

DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBB	UUU	UUU	GGGGGGGGGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBB	UUU	UUU	GGGGGGGGGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBB	UUU	UUU	GGGGGGGGGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDD	DDD	EEE	UUU	UUU	GGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBB	UUUUUUUUUUUUUU	UUUUUUUUUUUUUU	GGGGGGGGGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBB	UUUUUUUUUUUUUU	UUUUUUUUUUUUUU	GGGGGGGGGG
DDDDDDDDDDDD	EEEEEEEEEEEEEE	BBBBBBBBBBBB	UUUUUUUUUUUUUU	UUUUUUUUUUUUUU	GGGGGGGGGG



```
1 0001 0 MODULE DBGDPC ( IDENT = 'V04-000' ) =
2 0002 1 BEGIN
3 0003 1
4 0004 1
5 0005 1 *****
6 0006 1 *
7 0007 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
8 0008 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
9 0009 1 * ALL RIGHTS RESERVED.
10 0010 1 *
11 0011 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
12 0012 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
13 0013 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
14 0014 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
15 0015 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
16 0016 1 * TRANSFERRED.
17 0017 1 *
18 0018 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
19 0019 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
20 0020 1 * CORPORATION.
21 0021 1 *
22 0022 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
23 0023 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
24 0024 1 *
25 0025 1 *
26 0026 1 *****
27 0027 1
28 0028 1
29 0029 1 ++
30 0030 1 FACILITY:      DEBUG (DBG)
31 0031 1
32 0032 1 ABSTRACT:
33 0033 1     Analyzes PC correlation tables for DEBUG.
34 0034 1
35 0035 1 ENVIRONMENT:  VAX/VMS, user mode, interrupts disabled.
36 0036 1
37 0037 1 AUTHOR:       Carol Peters, CREATION DATE:   16 September 1977
38 0038 1
39 0039 1 Version      3.01
40 0040 1
41 0041 1 MODIFIED BY:
42 0042 1     (PS = Ping Sager, RT = Rich Title, JF = John Francis)
43 0043 1
44 0044 1 3.01 15-Sep-81    PS    Correct LINE END PC address calculation in
45 0045 1                    PC_TO_LINE_LOOKUP.
46 0046 1 3.02 23-Apr-82    RT    Fixed a bug in DBG$PC_TO_LINE_LOOKUP: the routine
47 0047 1                    was assuming that chasing upscope pointers will
48 0048 1                    always get you to a routine RST entry.
49 0049 1 4.0  13-Dec-82    PS    Switched some old symbolization routines to
50 0050 1                    use new code.
51 0051 1                    1-Mar-83    JF    Changed return values from DBG$PC_TO_LINE_LOOKUP
52 0052 1                    so that SUCCESS and FAILURE are shown properly
53 0053 1                    12-Apr-83    RT    Fixed a bug in PC_TO_LINE
54 0054 1                    24-Dec-83    RT    Added comments and did some general cleanup
55 0055 1 --
```

```
57 0056 1 ! TABLE OF CONTENTS:
58 0057 1 !
59 0058 1 ! FORWARD ROUTINE
60 0059 1 dbg$line_to_pc_lookup, ! Given line number associated it to a PC
61 0060 1 dbg$pc_to_line, ! Matches a PC to a line number
62 0061 1 dbg$pc_to_line_lookup, ! Given PC looks up associated line number
63 0062 1 proc_pc_cmd, ! Processes a string of PC correlation commands
64 0063 1 find_eot, ! Find end of line
65 0064 1 give_line_info: NOVALUE; ! Give more info about line number
66 0065 1
67 0066 1
68 0067 1 ! INCLUDE FILES:
69 0068 1 !
70 0069 1 REQUIRE 'SRC$:DBGPROLOG.REQ';
71 0203 1 LIBRARY 'LIB$:DBGGEN.L32';
72 0204 1
73 0205 1
74 0206 1 ! MACROS:
75 0207 1 !
76 0208 1 MACRO
77 0209 1 current_byte = 0, 0, 8, 1%, ! current top of record
78 0210 1 next_uns_byte = 1, 0, 8, 0%, ! byte argument to command
79 0211 1 next_uns_word = 1, 0, 16, 0%, ! word argument to command
80 0212 1 next_uns_long = 1, 0, 32, 0%, ! longword argument to command
81 0213 1 add_one_byte = 1, 0, 8, 0%, ! increment for top of record
82 0214 1 add_two_bytes = 2, 0, 8, 0%, ! ditto
83 0215 1 add_three_bytes = 3, 0, 8, 0%, ! ditto
84 0216 1 add_five_bytes = 5, 0, 8, 0%, ! ditto
85 0217 1
86 0218 1
87 0219 1 ! EQUATED SYMBOLS:
88 0220 1 !
89 0221 1 LITERAL
90 0222 1 line_open = 1,
91 0223 1 line_closed = 2;
92 0224 1
93 0225 1
94 0226 1 ! OWN STORAGE:
95 0227 1 !
96 0228 1 OWN
97 0229 1 dst_entry : REF dst$record,
98 0230 1 dpc_entry : REF BLOCK [, BYTE],
99 0231 1 start_pc,
100 0232 1 current_line,
101 0233 1 current_stmt,
102 0234 1 current_incr,
103 0235 1 current_pc,
104 0236 1 current_stmt_mode,
105 0237 1 current_mark,
106 0238 1 prev_line,
107 0239 1 prev_stmt,
108 0240 1 prev_incr,
109 0241 1 prev_pc,
110 0242 1 prev_stmt_mode,
111 0243 1 prev_mark,
112 0244 1 NUM_PC_TBLS, ! The number of PC-Correlation DST
113 0245 1 ! records for the current module
```

DBGDPC  
V04-000

K 15  
16-Sep-1984 00:22:28  
14-Sep-1984 12:16:51

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[DEBUG.SRC]DBGDPC.B32;1 Page 3 (2)

```
: 114      0246 1      current_table,
: 115      0247 1      report_next_line,
: 116      0248 1      report_next_stmt,
: 117      0249 1      report_prev_line,
: 118      0250 1      report_prev_stmt,
: 119      0251 1      pctbl_count;
: 120      0252 1
: 121      0253 1
: 122      0254 1      ! EXTERNAL REFERENCES:
: 123      0255 1      !
: 124      0256 1      EXTERNAL ROUTINE
: 125      0257 1      dbg$format_fao out: NOVALUE,      ! Forward FAO string
: 126      0258 1      dbg$pc_to_symid;                  ! Search Moudle SAT to locate RST
: 127      0259 1
```

```

129 0260 1 ROUTINE dbg$pc_to_line (match_pc_ptr, modpctbl,   pctbl_base,
130 0261 1                               line_no_ptr, stmt_no_ptr, line_pc) =
131 0262 1
132 0263 1 ++
133 0264 1 FUNCTIONAL DESCRIPTION:
134 0265 1
135 0266 1     This routine matches an address to a line number.
136 0267 1     The caller, DBG$PC_TO_LINE_LOOKUP, does the work of finding
137 0268 1     the PC/LINE table for the module containing the address.
138 0269 1     A pointer to this table is passed to this routine.
139 0270 1
140 0271 1     Each PC correlation record that exists for the module
141 0272 1     is sequentially analyzed until the desired address
142 0273 1     is seen.
143 0274 1
144 0275 1     See the comments in DBG$PC_TO_LINE_LOOKUP for more details
145 0276 1     about how this routine is used.
146 0277 1
147 0278 1 FORMAL PARAMETERS:
148 0279 1
149 0280 1     match_pc_ptr - The address to be matched.
150 0281 1     modpctbl     - The address of the table of pointers to
151 0282 1                   PC/LINE tables in this module. The first
152 0283 1                   longword of the table is a count of PC/LINE
153 0284 1                   tables, and the remaining longwords are
154 0285 1                   pointers to the DST records containing the tables.
155 0286 1
156 0287 1     pctbl_base   - The address which is the base address for
157 0288 1                   the PC/LINE tables
158 0289 1
159 0290 1     line_no_ptr  - An output parameter for the line number.
160 0291 1     stmt_no_ptr  - An output parameter for the statement number.
161 0292 1     line_pc      - An output parameter for the start pc of the
162 0293 1                   selected line/stmt.
163 0294 1
164 0295 1 ROUTINE VALUE:
165 0296 1
166 0297 1     This routine returns one of three values: 0, 1, or 2.
167 0298 1     Note that the caller, DBG$PC_TO_LINE_LOOKUP, may change
168 0299 1     return status "1" to return status "3" if we did not get
169 0300 1     an exact match. See that routine for further details
170 0301 1     on how the return status is used.
171 0302 1
172 0303 1     0 - If no match can be made because pc/line tables are
173 0304 1         not available for the given address. This may occur
174 0305 1         because the module containing the address was not
175 0306 1         set or was compiled /NODEBUG, or because the address
176 0307 1         is in system space or in an RTL shareable image.
177 0308 1
178 0309 1     1 - If a line number/stmt number was found.
179 0310 1
180 0311 1     2 - If there are pc/line tables available for the
181 0312 1         module containing the given address, but no match
182 0313 1         was found. This occurs if the address is not within
183 0314 1         any line in the module. The use of the "TERM" record
184 0315 1         in PC/LINE tables terminates an address range for
185 0316 1         a line without starting a new line, and this can
186 0317 1         give rise to addresses without line numbers.
187 0318 1
188 0319 1 --
189 0320 1 BEGIN
190 0321 1 MAP
191 0322 1 MODPCTBL: REF VECTOR[,LONG];
```

```
186 0317 2
187 0318 2
188 0319 2
189 0320 2
190 0321 2
191 0322 2
192 0323 2
193 0324 2
194 0325 2
195 0326 2
196 0327 2
197 0328 2
198 0329 2
199 0330 2
200 0331 2
201 0332 2
202 0333 2
203 0334 2
204 0335 2
205 0336 2
206 0337 2
207 0338 2
208 0339 2
209 0340 2
210 0341 2
211 0342 2
212 0343 2
213 0344 2
214 0345 2
215 0346 2
216 0347 2
217 0348 2
218 0349 2
219 0350 2
220 0351 2
221 0352 2
222 0353 2
223 0354 2
224 0355 2
225 0356 2
226 0357 2
227 0358 2
228 0359 2
229 0360 2
230 0361 2
231 0362 2
232 0363 2
233 0364 2
234 0365 2
235 0366 2
236 0367 2
237 0368 2
238 0369 2
239 0370 2
240 0371 2
241 0372 2
242 0373 2

! If we do not have a PC/LINE table, just return 0.
IF .MODPCTBL EQL 0 THEN RETURN 0;

! Set up the OWN variables that we use for reading the PC/LINE tables.
! This includes a count of the number of PC/LINE DST records in this
! module we have looked at so far (initialized to 1 here), a count
! of the total number of PC/LINE DST records in the module, a pointer
! to our position in the table of PC/LINE DST records,
! and a pointer to the first such DST record.
! If there are zero PC/LINE tables in this module, return 0 here.
PCTBL_COUNT = 1;
NUM_PC_TBLS = .MODPCTBL[0];
CURRENT_TABLE = MODPCTBL[1];
DST_ENTRY = .MODPCTBL[1];
IF .NUM_PC_TBLS EQL 0 THEN RETURN 0;

! Initialize the state variables (OWN variables in this module)
! that are used by PROC_PC_CMD.
current_line = 0;
current_stmt = 1;
current_incr = 1;
current_stmt_mode = FALSE;
current_pc = start_pc = .pctbl_base;
current_mark = line_closed;

! Call a routine that processes all PC correlation commands
! until a delta-PC command is seen. Then process that
! delta-PC command and return to this routine. If the processing
! is generally successful, return 1, otherwise return 0.
dpc_entry = dst_entry [dst$b_vflags];
REPEAT
BEGIN
prev_line = .current_line;
prev_stmt = .current_stmt;
prev_incr = .current_incr;
prev_stmt_mode = .current_stmt_mode;
prev_pc = .current_pc;
prev_mark = .current_mark;

! If we PROC_PC_CMD fails we have come to the end
! of the PC/LINE table for this module, without finding
! a match. In this case, return 2, indicating that we
! are in a module with PC/LINE tables, but we could not
! match the given PC.
IF NOT proc_pc_cmd ( )
THEN
```

```

: 243      0374      3      RETURN 2;
: 244      0375      3
: 245      0376      3
: 246      0377      3
: 247      0378      3      ! Report a match to a line if:
: 248      0379      3      - the PC is within the range given by
: 249      0380      3      the previous PC and the current PC, and
: 250      0381      3      - the line is marked as being OPEN.
: 251      0382      3
: 252      0383      3      IF (.prev_pc LEQA .match_pc_ptr) AND
: 253      0384      4      (.match_pc_ptr LSSA .current_pc) AND
: 254      0385      3      (.prev_mark EQL line_open)
: 255      0386      4      THEN
: 256      0387      3          BEGIN
: 257      0388      4              .stmt_no_ptr = (IF .prev_stmt EQL 1 THEN 0
: 258      0389      4                  ELSE .prev_stmt); ! Huh?
: 259      0390      4              .line_no_ptr = .prev_line;
: 260      0391      4              .line_pc = .prev_pc;
: 261      0392      3              RETURN 1;
: 262      0393      3              END;
: 263      0394      3
: 264      0395      3      ! Found nothing this round; continue trying.
: 265      0396      3
: 266      0397      3      END;
: 267      0398      3      ! End of REPEAT.
: 268      0399      3
: 269      0400      3      ! We have not found a match - return 2, indicating that we
: 270      0401      3      are in a module with PC/LINE tables, but we could not
: 271      0402      3      match the given PC.
: 272      0403      3
: 273      0404      3      RETURN 2;
: 274      0405      1      END;

```

```

.TITLE  DBGDPC
.IDENT  \V04-000\

.PSECT  DBG$OWN,NOEXE, PIC,2

00000 DST_ENTRY:
      .BLKB  4
00004 DPC_ENTRY:
      .BLKB  4
00008 START_PC:
      .BLKB  4
0000C CURRENT_LINE:
      .BLKB  4
00010 CURRENT_STMT:
      .BLKB  4
00014 CURRENT_INCR:
      .BLKB  4
00018 CURRENT_PC:
      .BLKB  4
0001C CURRENT_STMT_MODE:
      .BLKB  4
00020 CURRENT_MARK:
      .BLKB  4

```

00024 PREV\_LINE:  
          .BLKB 4  
00028 PREV\_STMT:  
          .BLKB 4  
0002C PREV\_INCR:  
          .BLKB 4  
00030 PREV\_PC:.BLKB 4  
00034 PREV\_STMT\_MODE:  
          .BLKB 4  
00038 PREV\_MARK:  
          .BLKB 4  
0003C NUM\_PC\_TBLS:  
          .BLKB 4  
00040 CURRENT\_TABLE:  
          .BLKB 4  
00044 REPORT\_NEXT\_LINE:  
          .BLRB 4  
00048 REPORT\_NEXT\_STMT:  
          .BLRB 4  
0004C REPORT\_PREV\_LINE:  
          .BLRB 4  
00050 REPORT\_PREV\_STMT:  
          .BLRB 4  
00054 PCTBL\_COUNT:  
          .BLKB 4

.EXTRN DBG\$FORMAT\_FAO\_OUT  
.EXTRN DBG\$PC\_TO\_SYMID

.PSECT DBG\$CODE,NOWRT, SHR, PIC,0

0004 00000 DBG\$PC\_TO LINE:

	52	00000000'	37	9E	00002	.WORD	Save R2	:	0260
	50	08	AC	D0	00009	MOVAB	NUM_PC_TBLS, R2	:	0321
			13	13	0000D	BEQL	1\$	:	
18	A2		01	D0	0000F	MOVL	#1, PCTBL_COUNT	:	0332
	62		60	D0	00013	MOVL	(R0), NUM_PC_TBLS	:	0333
04	A2	04	A0	9E	00016	MOVAB	4(R0), CURRENT_TABLE	:	0334
C4	A2	04	A0	D0	0001B	MOVL	4(R0), DST_ENTRY	:	0335
			62	D5	00020	TSTL	NUM_PC_TBLS	:	0336
			73	13	00022	1\$: BEQL	6\$	:	
		D0	A2	D4	00024	CLRL	CURRENT_LINE	:	0342
D4	A2		01	D0	00027	MOVL	#1, CURRENT_STMT	:	0343
D8	A2		01	D0	0002B	MOVL	#1, CURRENT_INCR	:	0344
		E0	A2	D4	0002F	CLRL	CURRENT_STMT_MODE	:	0345
	50	0C	AC	D0	00032	MOVL	PCTBL_BASE, R0	:	0346
CC	A2		50	D0	00036	MOVL	R0, START_PC	:	
DC	A2		50	D0	0003A	MOVL	R0, CURRENT_PC	:	
E4	A2		02	D0	0003E	MOVL	#2, CURRENT_MARK	:	0347
C8	A2		02	C1	00042	ADDL3	#2, DST_ENTRY, DPC_ENTRY	:	0355
E8	A2	D0	A2	7D	00048	2\$: MOVQ	CURRENT_LINE, PREV_LINE	:	0358
F8	A2	E0	A2	7D	0004D	MOVQ	CURRENT_STMT_MODE, PREV_STMT_MODE	:	0361
F0	A2	D8	A2	7D	00052	MOVQ	CURRENT_INCR, PREV_INCR	:	0360
0000V	CF		00	FB	00057	CALLS	#0, PROC_PC_CMD	:	0372
	34		50	E9	0005C	BLBC	R0, 5\$	:	
04	AC	F4	A2	D1	0005F	CMPL	PREV_PC, MATCH_PC_PTR	:	0382

DC	A2	04	E2	1A	00064	BGTRU	2\$	:	0383
			AC	D1	00066	CMPL	MATCH_PC_PTR, CURRENT_PC	:	
	01	FC	DB	1E	0006B	BGEQU	2\$	:	0384
			A2	D1	0006D	CMPL	PREV_MARK, #1	:	
	01	EC	D5	12	00071	BNEQ	2\$	:	0387
			A2	D1	00073	CMPL	PREV_STMT, #1	:	
			04	12	00077	BNEQ	3\$	:	
			50	D4	00079	CLRL	R0	:	
			04	11	0007B	BRB	4\$	:	
	50	EC	A2	D0	0007D	MOVL	PREV_STMT, R0	:	0388
14	BC		50	D0	00081	MOVL	R0, @STMT_NO_PTR	:	0387
10	BC	E8	A2	D0	00085	MOVL	PREV_LINE, @LINE_NO_PTR	:	0389
18	BC	F4	A2	D0	0008A	MOVL	PREV_PC, @LINE_PC	:	0390
	50		01	D0	0008F	MOVL	#1, R0	:	0391
				04	00092	RET		:	
	50		02	D0	00093	MOVL	#2, R0	:	0404
				04	00096	RET		:	
			50	D4	00097	CLRL	R0	:	0405
			04	00099	RET			:	

; Routine Size: 154 bytes, Routine Base: DBG\$CODE + 0000

```
: 276 0406 1 GLOBAL ROUTINE DBG$LINE_TO_PC_LOOKUP (LINE_NUM, STMT_NUM, MC_PTR,  
: 277 0407 1 LINE_PC, LINE_END, FLAG) =  
: 278 0408 1  
: 279 0409 1 FUNCTIONAL DESCRIPTION:  
: 280 0410 1 This routine finds the absolute PC address associated with  
: 281 0411 1 a line number/statement number.  
: 282 0412 1  
: 283 0413 1 Each PC correlation record that exists for a single routine  
: 284 0414 1 is sequentially analyzed until the desired line number  
: 285 0415 1 is seen.  
: 286 0416 1  
: 287 0417 1 If a match cannot be made because an end of routine record or  
: 288 0418 1 an invalid record is recognized, then this routine returns  
: 289 0419 1 FALSE.  
: 290 0420 1  
: 291 0421 1 FORMAL PARAMETERS:  
: 292 0422 1 line_num - the line number to find.  
: 293 0423 1 stmt_num - the statement number to find.  
: 294 0424 1 mc_ptr - module rstptr  
: 295 0425 1 line_pc - where to store the computed address.  
: 296 0426 1 line_end - a copy-back pointer for the line-end pc value.  
: 297 0427 1 flag - flag set to indicate more line information is needed.  
: 298 0428 1  
: 299 0429 1 ROUTINE VALUE:  
: 300 0430 1 The routine value is TRUE if the desired line was successfully  
: 301 0431 1 found; it is FALSE otherwise.  
: 302 0432 1  
: 303 0433 1  
: 304 0434 1  
: 305 0435 2 BEGIN  
: 306 0436 2 MAP  
: 307 0437 2 MC_PTR: REF RST$ENTRY;  
: 308 0438 2  
: 309 0439 2 LOCAL  
: 310 0440 2 MODPCTBL: REF VECTOR[,LONG];  
: 311 0441 2  
: 312 0442 2  
: 313 0443 2 ! Adjust a statement number of 1 to 0 (%LINE 10.1 is equivalent  
: 314 0444 2 ! to %LINE 10, and the algorithm below coughs at statement numbers of 1  
: 315 0445 2  
: 316 0446 2 IF .STMT_NUM EQL 1 THEN STMT_NUM = 0;  
: 317 0447 2  
: 318 0448 2  
: 319 0449 2 ! Set up the OWN variables that we use for reading the PC/LINE tables.  
: 320 0450 2 ! This includes a count of the number of PC/LINE DST records in this  
: 321 0451 2 ! module we have looked at so far (initialized to 1 here), a count  
: 322 0452 2 ! of the total number of PC/LINE DST records in the module, a pointer  
: 323 0453 2 ! to our position in the table of PC/LINE DST records,  
: 324 0454 2 ! and a pointer to the first such DST record.  
: 325 0455 2 ! If there are zero PC/LINE tables in this module, return 0 here.  
: 326 0456 2  
: 327 0457 2 PCTBL COUNT = 1;  
: 328 0458 2 MODPCTBL = .MC_PTR[RST$ MODPCTBL];  
: 329 0459 2 IF .MODPCTBL EQL 0 THEN RETURN FALSE;  
: 330 0460 2 NUM_PC_TBLS = .MODPCTBL[0];  
: 331 0461 2 CURRENT_TABLE = MODPCTBL[1];  
: 332 0462 2 DST_ENTRY = .MODPCTBL[1];
```

```

333 0463 2 IF .NUM_PC_TBLS EQL 0 THEN RETURN 0;
334 0464 2
335 0465 2
336 0466 2 ! Initialize state variables. These are OWN variables that
337 0467 2 ! are used by PROC_PC_CMD.
338 0468 2
339 0469 2 current_line = 0;
340 0470 2 current_stmt = 1;
341 0471 2 current_incr = 1;
342 0472 2 current_stmt_mode = FALSE;
343 0473 2 current_pc = start_pc = .mc_ptr[rst$l_pctbl_base];
344 0474 2 current_mark = line_closed;
345 0475 2
346 0476 2
347 0477 2 ! Loop through the PC Correlation Tables for this module until the
348 0478 2 ! desired line number is found or the table ends. To do this, we call
349 0479 2 ! PROC_PC_CMD to process all PC Correlation commands until a delta-PC
350 0480 2 ! command is found. It then returns a PC and a line number and we
351 0481 2 ! check whether that is the line number we are looking for. If not,
352 0482 2 ! we loop for the next line until the desired line is found or no PC
353 0483 2 ! Correlation commands remain.
354 0484 2
355 0485 2 dpc_entry = dst_entry [dst$b_vflags];
356 0486 2 REPORT_PREV_LINE = 0;
357 0487 2 REPORT_PREV_STMT = 1;
358 0488 2 REPORT_NEXT_LINE = .LINE_NUM;
359 0489 2 REPORT_NEXT_STMT = .STMT_NUM;
360 0490 2 WHILE TRUE DO
361 0491 2 BEGIN
362 0492 2
363 0493 2 ! Remember the previous values of all the state variables
364 0494 2 ! before getting the current values this time around.
365 0495 2
366 0496 2
367 0497 2 PREV_LINE = .CURRENT_LINE;
368 0498 2 PREV_STMT = .CURRENT_STMT;
369 0499 2 PREV_INCR = .CURRENT_INCR;
370 0500 2 PREV_STMT_MODE = .CURRENT_STMT_MODE;
371 0501 2 PREV_PC = .CURRENT_PC;
372 0502 2 PREV_MARK = .CURRENT_MARK;
373 0503 2
374 0504 2
375 0505 2 ! Call PROC_PC_CMD to get the next PC - line number pair.
376 0506 2 ! When there are no more lines, exit this loop.
377 0507 2
378 0508 2 IF NOT PROC_PC_CMD() THEN EXITLOOP;
379 0509 2
380 0510 2
381 0511 2 ! Set report next line and stmt for the first time.
382 0512 2
383 0513 2 IF (.REPORT_NEXT_LINE EQL .LINE_NUM) AND
384 0514 2 (.REPORT_NEXT_STMT EQL .STMT_NUM)
385 0515 2 THEN
386 0516 2 BEGIN
387 0517 2 IF (.CURRENT_LINE GTR .LINE_NUM) OR
388 0518 2 ((.CURRENT_LINE EQL .LINE_NUM) AND
389 0519 2 (.CURRENT_STMT GTR .STMT_NUM))
```

```
THEN
  BEGIN
    REPORT_NEXT_LINE = .CURRENT_LINE;
    REPORT_NEXT_STMT = .CURRENT_STMT;
  END;

END;

! At this point we have Prev. line, current line, and given line info.
! So we define the reporting line information centered around given line.
! (we choose the closest two ends value).
! Define report prev. line.
IF .REPORT_PREV_LINE LSS .LINE_NUM
THEN
  BEGIN
    IF .PREV_LINE LSS .LINE_NUM
    THEN
      REPORT_PREV_LINE = MAX(.REPORT_PREV_LINE, .PREV_LINE)
    ELSE
      BEGIN
        IF ((.PREV_LINE EQL .LINE_NUM) AND
            (.PREV_STMT LSS .STMT_NUM))
        THEN
          BEGIN
            REPORT_PREV_LINE = .PREV_LINE;
            REPORT_PREV_STMT = .PREV_STMT;
          END;
        END;
      END
    END
  ELSE
    BEGIN
      IF ((.REPORT_PREV_LINE EQL .LINE_NUM) AND
          (.REPORT_PREV_STMT LSS .STMT_NUM))
      THEN
        BEGIN
          IF (.PREV_LINE EQL .LINE_NUM) AND
              (.PREV_STMT LSS .STMT_NUM)
          THEN
            REPORT_PREV_STMT = MAX(.PREV_STMT, .REPORT_PREV_STMT);
          END;
        END;
      END;
    END;
  END;

! Define report next line.
IF .REPORT_NEXT_LINE GTR .LINE_NUM
THEN
  BEGIN
    IF .CURRENT_LINE GTR .LINE_NUM
```

```

447      0577 4
448      0578 4
449      0579 4
450      0580 5
451      0581 6
452      0582 6
453      0583 5
454      0584 6
455      0585 6
456      0586 6
457      0587 5
458      0588 5
459      0589 4
460      0590 4
461      0591 4
462      0592 4
463      0593 3
464      0594 4
465      0595 5
466      0596 5
467      0597 4
468      0598 5
469      0599 5
470      0600 6
471      0601 5
472      0602 5
473      0603 5
474      0604 4
475      0605 4
476      0606 3
477      0607 3
478      0608 3
479      0609 3
480      0610 3
481      0611 3
482      0612 3
483      0613 3
484      0614 3
485      0615 3
486      0616 3
487      0617 3
488      0618 3
489      0619 3
490      0620 4
491      0621 4
492      0622 4
493      0623 3
494      0624 5
495      0625 5
496      0626 5
497      0627 6
498      0628 6
499      0629 6
500      0630 5
501      0631 5
502      0632 5
503      0633 5

      THEN
      REPORT_NEXT_LINE = MIN(.REPORT_NEXT_LINE, .CURRENT_LINE)
    ELSE
      BEGIN
      IF ((.CURRENT_LINE EQL .LINE_NUM) AND
      (.CURRENT_STMT GTR .STMT_NUM))
      THEN
      BEGIN
      REPORT_NEXT_LINE = .CURRENT_LINE;
      REPORT_NEXT_STMT = .CURRENT_STMT;
      END;
      END;
    END
  ELSE
    BEGIN
    IF ((.REPORT_NEXT_LINE EQL .LINE_NUM) AND
    (.REPORT_NEXT_STMT GTR .STMT_NUM))
    THEN
    BEGIN
    IF ((.CURRENT_LINE EQL .LINE_NUM) AND
    (.CURRENT_STMT GTR .STMT_NUM))
    THEN
      REPORT_PREV_STMT = MIN(.CURRENT_STMT, .REPORT_NEXT_STMT);
    END;
    END;
  END;

  ! Note that: above code did not take care of the equality condition.
  ! it should be set up here and tested in give_line_info.

  ! If the current line number is equivalent to the one we were
  ! passed (and this includes the statement number), then we
  ! return the corresponding PC to LINE_PC and we return TRUE.
  ! If we are at the right line but there is no such statement
  ! number, we clear LINE_PC and return FALSE.
  IF .CURRENT_LINE EQL .LINE_NUM
  THEN
    BEGIN
    IF MAX (.CURRENT_STMT, 1) EQL MAX(.STMT_NUM, 1)
    THEN
      BEGIN
      .LINE_PC = .CURRENT_PC;
      IF NOT FIND_EOL(.LINE_END)
      THEN
      BEGIN
      IF .FLAG THEN GIVE_LINE_INFO(.LINE_NUM, .STMT_NUM);
      RETURN FALSE;
      END;
      RETURN TRUE;
    END
  END
```

```
504 0634 5
505 0635 4
506 0636 5
507 0637 5
508 0638 5
509 0639 6
510 0640 6
511 0641 6
512 0642 6
513 0643 5
514 0644 5
515 0645 4
516 0646 4
517 0647 4
518 0648 3
519 0649 2
520 0650 2
521 0651 2
522 0652 2
523 0653 2
524 0654 2
525 0655 2
526 0656 2
527 0657 2
528 0658 1

ELSE
BEGIN
IF MAX(.CURRENT_STMT,1) GTR MAX(.STMT_NUM,1)
THEN
BEGIN
.LINE_PC = 0;
IF .FLAG THEN GIVE_LINE_INFO(.LINE_NUM, .STMT_NUM);
RETURN FALSE;
END;
END;
END;

END;

! End of WHILE loop over PC Corr Tbl

! The desired line number was not found. Clear LINE_PC and return FALSE
as the routine value.

IF .FLAG THEN GIVE_LINE_INFO(.LINE_NUM, .STMT_NUM);
.LINE_PC = 0;
RETURN FALSE;
END;
```

54	00000000	001C	000000	.ENTRY	DBG\$LINE TO PC LOOKUP, Save R2,R3,R4	0406
01	08	EF	9E 000002	MOVAB	CURRENT_STMT, R4	0446
		AC	D1 000009	CMPL	STMT_NUM, #1	
		03	12 00000D	BNEQ	1\$	
44	A4	08	AC D4 00000F	CLRL	STMT_NUM	0457
50	0C	01	D0 00012	MOVL	#1, PCTBL COUNT	0458
51	2C	AC	D0 00016	MOVL	MC_PTR, R0	
		A0	D0 0001A	MOVL	44(R0), MODPCTBL	
		11	13 0001E	BEQL	2\$	0459
2C	A4	61	D0 00020	MOVL	(MODPCTBL), NUM_PC_TBLS	0460
30	A4	04	A1 9E 00024	MOVAB	4(MODPCTBL), CURRENT_TABLE	0461
F0	A4	04	A1 D0 00029	MOVL	4(MODPCTBL), DST_ENTRY	0462
		2C	A4 D5 0002E	TSTL	NUM_PC_TBLS	0463
		03	12 00031	BNEQ	3\$	
		0170	31 00033	BRW	26\$	
		FC	A4 D4 00036	CLRL	CURRENT_LINE	0469
	64	01	D0 00039	MOVL	#1, CURRENT_STMT	0470
04	A4	01	D0 0003C	MOVL	#1, CURRENT_INCR	0471
		0C	A4 D4 00040	CLRL	CURRENT_STMT_MODE	0472
	50	1C	A0 D0 00043	MOVL	28(R0), R0	0473
F8	A4	50	D0 00047	MOVL	R0, START_PC	
08	A4	50	D0 0004B	MOVL	R0, CURRENT_PC	
10	A4	02	D0 0004F	MOVL	#2, CURRENT_MARK	0474
F4	A4	02	C1 00053	ADDL3	#2, DST_ENTRY, DPC_ENTRY	0485
		3C	A4 D4 00059	CLRL	REPORT_PREV_LINE	0486
40	A4	01	D0 0005C	MOVL	#1, REPORT_PREV_STMT	0487
52	04	AC	7D 00060	MOVQ	LINE_NUM, R2	0488

34	A4	52	7D	00064	MOVQ	R2, REPORT_NEXT_LINE	:	
14	A4	FC	A4	7D	00068	4\$: MOVQ	CURRENT_LINE, PREV_LINE	0497
24	A4	OC	A4	7D	0006D	MOVQ	CURRENT_STMT_MODE, PREV_STMT_MODE	0500
1C	A4	04	A4	7D	00072	MOVQ	CURRENT_INCR, PREV_INCR	0499
0000V	CF	00	FB	00077	CALLS	70, PROC_PC_CMD	:	0508
	03	50	E8	0007C	BLBS	R0, 5\$	:	
		0116	31	0007F	BRW	24\$	:	
	52	34	A4	D1	00082	5\$: CMPL	REPORT_NEXT_LINE, R2	0513
			1E	12	00086	BNEQ	7\$	
	53	38	A4	D1	00088	CMPL	REPORT_NEXT_STMT, R3	0514
			18	12	0008C	BNEQ	7\$	
	50	FC	A4	D0	0008E	MOVL	CURRENT_LINE, R0	0517
	52		50	D1	00092	CMPL	R0, R2	
			07	14	00095	BGTR	6\$	
			0D	12	00097	BNEQ	7\$	0518
	53		64	D1	00099	CMPL	CURRENT_STMT, R3	0519
			08	15	0009C	BLEQ	7\$	
34	A4		50	D0	0009E	6\$: MOVL	R0, REPORT_NEXT_LINE	0522
38	A4		64	D0	000A2	MOVL	CURRENT_STMT, REPORT_NEXT_STMT	0523
	50	3C	A4	D0	000A6	7\$: MOVL	REPORT_PREV_LINE, R0	0535
	52		50	D1	000AA	CMPL	R0, R2	
			2A	18	000AD	BGEQ	10\$	
	51	14	A4	D0	000AF	MOVL	PREV_LINE, R1	0538
	52		51	D1	000B3	CMPL	R1, R2	
			0E	18	000B6	BGEQ	9\$	
	51		50	D1	000B8	CMPL	R0, R1	0540
			03	18	000BB	BGEQ	8\$	
	50		51	D0	000BD	MOVL	R1, R0	
3C	A4		50	D0	000C0	8\$: MOVL	R0, REPORT_PREV_LINE	
			39	11	000C4	BRB	12\$	
			37	12	000C6	9\$: BNEQ	12\$	0543
	53	18	A4	D1	000C8	CMPL	PREV_STMT, R3	0544
			31	18	000CC	BGEQ	12\$	
3C	A4		51	D0	000CE	MOVL	R1, REPORT_PREV_LINE	0547
40	A4	18	A4	D0	000D2	MOVL	PREV_STMT, REPORT_PREV_STMT	0548
			26	11	000D7	BRB	12\$	0535
			24	12	000D9	10\$: BNEQ	12\$	0557
	53	40	A4	D1	000DB	CMPL	REPORT_PREV_STMT, R3	0558
			1E	18	000DF	BGEQ	12\$	
	52	14	A4	D1	000E1	CMPL	PREV_LINE, R2	0561
			18	12	000E5	BNEQ	12\$	
	53	18	A4	D1	000E7	CMPL	PREV_STMT, R3	0562
			12	18	000EB	BGEQ	12\$	
	50	18	A4	D0	000ED	MOVL	PREV_STMT, R0	0564
40	A4		50	D1	000F1	CMPL	R0, REPORT_PREV_STMT	
			04	18	000F5	BGEQ	11\$	
	50	40	A4	D0	000F7	MOVL	REPORT_PREV_STMT, R0	
40	A4		50	D0	000FB	11\$: MOVL	R0, REPORT_PREV_STMT	
	50	34	A4	D0	000FF	12\$: MOVL	REPORT_NEXT_LINE, R0	0573
	52		50	D1	00103	CMPL	R0, R2	
			28	15	00106	BLEQ	15\$	
	51	FC	A4	D0	00108	MOVL	CURRENT_LINE, R1	0576
	52		51	D1	0010C	CMPL	R1, R2	
			0E	15	0010F	BLEQ	14\$	
	51		50	D1	00111	CMPL	R0, R1	0578
			03	15	00114	BLEQ	13\$	
	50		51	D0	00116	MOVL	R1, R0	

34	A4		50	D0	00119	13\$:	MOVL	R0	REPORT_NEXT_LINE	:	
			35	11	0011D		BRB	17\$		:	0581
	53		33	12	0011F	14\$:	BNEQ	17\$		:	0582
			64	D1	00121		CMPL	CURRENT_STMT, R3		:	
			2E	15	00124		BLEQ	17\$		:	
34	A4		51	D0	00126		MOVL	R1, REPORT_NEXT_LINE		:	0585
38	A4		64	D0	0012A		MOVL	CURRENT_STMT, REPORT_NEXT_STMT		:	0586
			24	11	0012E		BRB	17\$		:	0573
			22	12	00130	15\$:	BNEQ	17\$		:	0595
	53	38	A4	D1	00132		CMPL	REPORT_NEXT_STMT, R3		:	0596
			1C	15	00136		BLEQ	17\$		:	
	52	FC	A4	D1	00138		CMPL	CURRENT_LINE, R2		:	0599
			16	12	0013C		BNEQ	17\$		:	
	53		64	D1	0013E		CMPL	CURRENT_STMT, R3		:	0600
			11	15	00141		BLEQ	17\$		:	
	50		64	D0	00143		MOVL	CURRENT_STMT, R0		:	0602
38	A4		50	D1	00146		CMPL	R0, REPORT_NEXT_STMT		:	
			04	15	0014A		BLEQ	16\$		:	
	50	38	A4	D0	0014C		MOVL	REPORT_NEXT_STMT, R0		:	
40	A4		50	D0	00150	16\$:	MOVL	R0, REPORT_PREV_STMT		:	
	52	FC	A4	D1	00154	17\$:	CMPL	CURRENT_LINE, R2		:	0618
			03	13	00158		BEQL	19\$		:	
		FF	0B	31	0015A	18\$:	BRW	4\$		:	
	51		64	D0	0015D	19\$:	MOVL	CURRENT_STMT, R1		:	0621
			03	14	00160		BGTR	20\$		:	
	51		01	D0	00162		MOVL	#1, R1		:	
	50		53	D0	00165	20\$:	MOVL	R3, R0		:	
			03	14	00168		BGTR	21\$		:	
	50		01	D0	0016A		MOVL	#1, R0		:	
	50		51	D1	0016D	21\$:	CMPL	R1, R0		:	
			14	12	00170		BNEQ	22\$		:	
10	BC	08	A4	D0	00172		MOVL	CURRENT_PC, @LINE_PC		:	0624
		14	AC	DD	00177		PUSHL	LINE_END		:	0625
0000V	CF		01	FB	0017A		CALLS	#1, FIND_EOL		:	
	09		50	E9	0017F		BLBC	R0, 23\$		:	
	50		01	D0	00182		MOVL	#1, R0		:	0632
			04	00185		RET				:	
			D2	15	00186	22\$:	BLEQ	18\$		:	0637
		10	BC	D4	00188		CLRL	@LINE_PC		:	0640
	17	18	AC	E9	0018B	23\$:	BLBC	FLAG, 26\$		:	0641
			0C	BB	0018F		PUSHR	#*M<R2,R3>		:	
0000V	CF		02	FB	00191		CALLS	#2, GIVE_LINE_INFO		:	
			0E	11	00196		BRB	26\$		:	0642
	07	18	AC	E9	00198	24\$:	BLBC	FLAG, 25\$		:	0655
			0C	BB	0019C		PUSHR	#*M<R2,R3>		:	
0000V	CF		02	FB	0019E		CALLS	#2, GIVE_LINE_INFO		:	
		10	BC	D4	001A3	25\$:	CLRL	@LINE_PC		:	0656
			50	D4	001A6	26\$:	CLRL	R0		:	0658
			04	001A8		RET				:	

; Routine Size: 425 bytes, Routine Base: DBG\$CODE + 009A

```
0659 1 GLOBAL ROUTINE dbg$pc_to_line_lookup (match_pc_ptr, line_no_ptr, stmt_no_ptr,  
0660 1 line_start, line_end, mod_symid) =  
0661 1  
0662 1 FUNCTIONAL DESCRIPTION:  
0663 1  
0664 1 This routine matches an address to a line number.  
0665 1 We need to do this in several situations:  
0666 1  
0667 1 1. When stepping by line, to determine when to stop stepping. (DBGEVENT)  
0668 1 2. When symbolizing a code address to put out "%LINE XX" (DBGSYMBLZ)  
0669 1 3. Putting out the SHOW CALLS display (DBGTBK)  
0670 1 4. Finding the start of the line for "EX/INS ^" (DBGLEVEL3)  
0671 1 5. Source display, as in EX/SOURCE .PC (DBGSOURCE)  
0672 1  
0673 1 The line number (and statement number, for BASIC) is returned.  
0674 1 Also returned are: the start and end address of the line,  
0675 1 and a pointer to the module RST entry for the module containing  
0676 1 the given address.  
0677 1  
0678 1 Each PC correlation record that exists for the module  
0679 1 is sequentially analyzed until the desired address is seen.  
0680 1  
0681 1 This routine is actually just a cover routine for DBG$PC_TO_LINE,  
0682 1 where the real work is done.  
0683 1  
0684 1 FORMAL PARAMETERS:  
0685 1  
0686 1 match_pc_ptr - the address to be matched.  
0687 1 line_no_ptr - an output parameter for the line number.  
0688 1 stmt_no_ptr - an output parameter for the statement number.  
0689 1 line_start - an output parameter for the start pc of the  
0690 1 selected line/stmt.  
0691 1 line_end - an output parameter for the end pc of the  
0692 1 selected line/stmt.  
0693 1 mod_symid - An in/out parameter, as follows:  
0694 1  
0695 1 If the caller has a SYMID for a block, routine,  
0696 1 or module which contains the given address, then  
0697 1 this symid can be passed in here. This saves  
0698 1 a search of the Static Address Table.  
0699 1 If the caller  
0700 1 does not have a symid, then zero is passed in.  
0701 1 Note that these are passed in with an extra level  
0702 1 of indirection, e.g.,  
0703 1 SYMID = 0;  
0704 1 STATUS = DBG$PC_TO_LINE_LOOKUP(.ADDRESS,....,SYMID);  
0705 1  
0706 1 In either case, this parameter is filled in with  
0707 1 the address of the module containing MATCH_PC_PTR.  
0708 1  
0709 1 ROUTINE VALUE:  
0710 1  
0711 1 This routine can return four values: 0, 1, 2, or 3.  
0712 1 Most of the callers just test the result for  
0713 1 TRUE (meaning a match was found), or FALSE (meaning a match  
0714 1 was not found). So for these callers, 0 and 2 are the same,  
0715 1 and 1 and 3 are the same.
```

```
587 0716 1
588 0717 1
589 0718 1
590 0719 1
591 0720 1
592 0721 1
593 0722 1
594 0723 1
595 0724 1
596 0725 1
597 0726 1
598 0727 1
599 0728 1
600 0729 1
601 0730 1
602 0731 1
603 0732 1
604 0733 1
605 0734 1
606 0735 1
607 0736 1
608 0737 1
609 0738 1
610 0739 1
611 0740 2
612 0741 2
613 0742 2
614 0743 2
615 0744 2
616 0745 2
617 0746 2
618 0747 2
619 0748 2
620 0749 2
621 0750 2
622 0751 2
623 0752 2
624 0753 2
625 0754 3
626 0755 3
627 0756 3
628 0757 3
629 0758 3
630 0759 3
631 0760 3
632 0761 3
633 0762 3
634 0763 3
635 0764 2
636 0765 2
637 0766 2
638 0767 2
639 0768 2
640 0769 2
641 0770 2
642 0771 2
643 0772 2
```

DBGEVENT needs more detailed information than just whether a match was found, in order to decide whether to continue stepping. It needs to know why a match was not found, or if one was found, whether or not it was an exact match. So for the DBGEVENT call, we return the following:

- 0 - If no match can be made because pc/line tables are not available for the given address. This may occur because the module containing the address was not set or was compiled /NODEBUG, or because the address is in system space or in an RTL shareable image.
- 1 - If a line number/stmt number was found, and we have an exact match to that line number.
- 2 - If there are pc/line tables available for the module containing the given address, but no match was found. This occurs if the address is not within any line in the module. The use of the "TERM" record in PC/LINE tables terminates an address range for a line without starting a new line, and this can give rise to addresses without line numbers.
- 3 - If there is a line number associated with the address, but it is not an exact match.

```
BEGIN
LOCAL
  rstptr: REF rst$entry,  ! Module RST pointer
  status;                  ! Return Status

! If we do not know an RST entry for a program unit
! containing the given address, we'll look
! it up through the Program-level SAT.
! If we already have the information
! (passed in from the caller) then just set it up.
IF ..mod_symid EQL 0
THEN
  BEGIN
    status = dbg$pc_to_symid(.match_pc_ptr, rstptr);

    ! If PC_TO_SYMID failed, then we do not have a module containing
    ! the address in our module chain. Thus, return zero.
    IF NOT .status THEN RETURN 0;
  END
ELSE
  rstptr = ..mod_symid;

! Go upscope to the module level, just in case a caller passed in
! a routine or block RST entry.
WHILE (.rstptr[rst$b_kind] NEQ rst$k_module) DO
  rstptr = .rstptr[rst$l_upscopeptr];
```

```

: 644      0773  2
: 645      0774  2
: 646      0775  2
: 647      0776  2
: 648      0777  2
: 649      0778  2
: 650      0779  2
: 651      0780  2
: 652      0781  2
: 653      0782  2
: 654      0783  2
: 655      0784  2
: 656      0785  2
: 657      0786  2
: 658      0787  2
: 659      0788  2
: 660      0789  2
: 661      0790  2
: 662      0791  2
: 663      0792  2
: 664      0793  2
: 665      0794  2
: 666      0795  2
: 667      0796  2
: 668      0797  3
: 669      0798  3
: 670      0799  3
: 671      0800  3
: 672      0801  3
: 673      0802  2
: 674      0803  2
: 675      0804  2
: 676      0805  1

! Set the return module RST.
.mod_symid = .rstptr;

! Now call the routine to do the real work. Pass along the three
! output parameters LINE_NO_PTR, STMT_NO_PTR, and LINE_START,
! to be filled in by DBG$PC_TO_LINE.
status = dbg$pc_to_line(.match_pc_ptr, .rstptr[rst$l_modpctbl],
                        .rstptr[rst$l_pctbl_base],
                        .line_no_ptr, .stmt_no_ptr, .line_start);

! We get the return code from DBG$PC_TO_LINE. Here we check
! for the PC being an exact match. If not, we change the "1"
! return status to a "3" to indicate this. We also fill in the
! LINE_END output parameter, using the OWN variable CURRENT_PC
! that gets set in the processing of PC/LINE records.
IF .status EQL 1
THEN
    BEGIN
        .line_end = .current_pc - 1;
        IF ..[line_start NEQA .match_pc_ptr
        THEN
            status = 3;          ! not exact match.
        END;
    RETURN .status;
END;
```

			0000	00000	.ENTRY	DBG\$PC_TO_LINE_LOOKUP, Save nothing	: 0659
	5E		04	C2 00002	SUBL2	#4, SP	: 0752
		18	BC	D5 00005	TSTL	@MOD_SYMID	: 0755
			11	12 00008	BNEQ	1\$	: 0761
			5E	DD 0000A	PUSHL	SP	: 0765
		04	AC	DD 0000C	PUSHL	MATCH_PC_PTR	: 0771
00000000G	00		02	FB 0000F	CALLS	#2, DBG\$PC_TO_SYMID	: 0772
	06		50	E8 00016	BLBS	STATUS, 2\$	: 0777
			48	11 00019	BRB	4\$	: 0786
	6E	18	BC	D0 0001B 1\$:	MOVL	@MOD_SYMID, RSTPTR	
	51		6E	D0 0001F 2\$:	MOVL	RSTPTR, R1	
	01	14	A1	91 00022	CMPB	20(R1), #1	
			06	13 00026	BEQL	3\$	
	6E	10	A1	D0 00028	MOVL	16(R1), RSTPTR	
			F1	11 0002C	BRB	2\$	
	51		6E	D0 0002E 3\$:	MOVL	RSTPTR, R1	
18	BC		51	D0 00031	MOVL	R1, @MOD_SYMID	
	7E	0C	AC	7D 00035	MOVQ	STMT_NO_PTR, -(SP)	
		08	AC	DD 00039	PUSHL	LINE_NO_PTR	

DBGDPC  
V04-000

B 1  
16-Sep-1984 00:22:28  
14-Sep-1984 12:16:51

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[DEBUG.SRC]DBGDPC.B32;1  
Page 19  
(5)

			1C	A1	DD	0003C	PUSHL	28(R1)	:	0785
			2C	A1	DD	0003F	PUSHL	44(R1)	:	0784
			04	AC	DD	00042	PUSHL	MATCH_PC_PTR	:	
	FD73	CF		06	FB	00045	CALLS	#6, DBG\$PC_TO_LINE	:	
		01		50	D1	0004A	CMPL	STATUS, #1	:	0795
				16	12	0004D	BNEQ	5\$	:	
14	BC 00000000'	EF		01	C3	0004F	SUBL3	#1, CURRENT_PC, @LINE_END	:	0798
	04	AC	10	BC	D1	00058	CMPL	@LINE_START, MATCH_PC_PTR	:	0799
				06	13	0005D	BEQL	5\$	:	
		50		03	D0	0005F	MOVL	#3, STATUS	:	0801
					04	00062	RET		:	0804
				50	D4	00063	CLRL	R0	:	0805
					04	00065	RET		:	
						4\$:			:	
						5\$:			:	

; Routine Size: 102 bytes, Routine Base: DBG\$CODE + 0243

```

678 0806 1 ROUTINE PROC_PC_CMD =
679 0807 1 ++
680 0808 1 Functional description:
681 0809 1
682 0810 1 This routine processes PC correlation commands until a
683 0811 1 delta-PC command is seen. The delta-PC command is also processed.
684 0812 1 Then this routine returns with all the contents of the
685 0813 1 parameter pointers updated appropriately.
686 0814 1
687 0815 1 This routine moves from PC record to PC record as necessary. If
688 0816 1 no more records are seen, this routine returns false. If
689 0817 1 an error is seen in a PC correlation record, then this
690 0818 1 routine sets the contents of line_ptr to zero and
691 0819 1 returns false.
692 0820 1
693 0821 1 Inputs:
694 0822 1
695 0823 1 Implicit inputs:
696 0824 1 None
697 0825 1
698 0826 1 Implicit outputs:
699 0827 1 the contents of the line pointer, the increment pointer, the
700 0828 1 statement pointer, the next_pc pointer, dpc_entry, and possible
701 0829 1 dst_entry are updated to new values.
702 0830 1
703 0831 1 Routine value:
704 0832 1 TRUE if all goes well, otherwise FALSE.
705 0833 1
706 0834 1 Side effects:
707 0835 1 More of the correlation records for this routine are read.
708 0836 1 --
709 0837 1
710 0838 2 BEGIN
711 0839 2
712 0840 2 REPEAT
713 0841 2 BEGIN
714 0842 2
715 0843 2
716 0844 2 ! See whether the current record is exhausted. If
717 0845 2 so, get a new record. If none are available,
718 0846 2 return FALSE. Otherwise, set dpc_entry to point to
719 0847 2 the address of the third byte of the correlation record.
720 0848 2
721 0849 2 IF dpc_entry[current_byte] GTR (.dst_entry[dst$b_length] +
722 0850 2 dst_entry[dst$b_length])
723 0851 2 THEN
724 0852 2 BEGIN
725 0853 2 PCTBL_COUNT = .PCTBL_COUNT + 1;
726 0854 2 IF .PCTBL_COUNT GTR .NUM_PC_TBL THEN RETURN FALSE;
727 0855 2 current_table = .current_table + 4;
728 0856 2 dst_entry = ..current_table;
729 0857 2 dpc_entry = dst_entry[dst$b_vflags];
730 0858 2 END;
731 0859 2
732 0860 2
733 0861 2 ! Now process each command, either PC correlation or
734 0862 2 ! delta-PC one at a time. Once a delta-PC command is
```

```
735 0863
736 0864
737 0865
738 0866
739 0867
740 0868
741 0869
742 0870
743 0871
744 0872
745 0873
746 0874
747 0875
748 0876
749 0877
750 0878
751 0879
752 0880
753 0881
754 0882
755 0883
756 0884
757 0885
758 0886
759 0887
760 0888
761 0889
762 0890
763 0891
764 0892
765 0893
766 0894
767 0895
768 0896
769 0897
770 0898
771 0899
772 0900
773 0901
774 0902
775 0903
776 0904
777 0905
778 0906
779 0907
780 0908
781 0909
782 0910
783 0911
784 0912
785 0913
786 0914
787 0915
788 0916
789 0917
790 0918
791 0919
```

```
! processed, control returns from this routine to its
! caller.
```

```
CASE .dpc_entry [current_byte] FROM 1 TO dst$sk_pccor_high OF
SET
```

```
! Read the next two bytes as an unsigned word
! representing a delta-PC value. Update the next_pc
! and update the dpc_entry address.
```

```
[dst$sk_delta_pc_w]:
BEGIN
IF .current_stmt_mode
THEN
current_stmt = .current_stmt + 1
ELSE
current_line = .current_line +
.current_incr;

current_mark = line_open;
current_pc = .current_pc +
.dpc_entry [next_uns_word];
dpc_entry = dpc_entry [add_three_bytes];
RETURN TRUE;
END;
```

```
! Read the next four bytes as an unsigned longword
! representing a delta-PC value. Update the next_pc
! and update the dpc_entry address.
```

```
[dst$sk_delta_pc_l]:
BEGIN
IF .current_stmt_mode
THEN
current_stmt = .current_stmt + 1
ELSE
current_line = .current_line +
.current_incr;

current_mark = line_open;
current_pc = .current_pc +
.dpc_entry [next_uns_long];
dpc_entry = dpc_entry [add_five_bytes];
RETURN TRUE;
END;
```

```
! Increase the current line number by the value
! contained in the next unsigned byte.
```

```
[dst$sk_incr_linum]:
BEGIN
current_line = .current_line + .dpc_entry [next_uns_byte];
IF .current_stmt_mode THEN current_stmt = 1;
dpc_entry = dpc_entry [add_two_bytes];
```

END;

! Increase the current line number by the value  
! contained in the next unsigned word.

[dst\$sk\_incr\_linum\_w]:

BEGIN

IF .current\_stmt\_mode THEN current\_stmt = 1;

current\_line = .current\_line + .dpc\_entry [next\_uns\_word];

dpc\_entry = dpc\_entry [add\_three\_bytes];

END;

! Increase the current line number by the value  
! contained in the next unsigned longword.

[dst\$sk\_incr\_linum\_l]:

BEGIN

IF .current\_stmt\_mode THEN current\_stmt = 1;

current\_line = .current\_line + .dpc\_entry [next\_uns\_long];

dpc\_entry = dpc\_entry [add\_five\_bytes];

END;

! Change the line increment from its present value to  
! the value contained in the next unsigned byte.

[dst\$sk\_set\_linum\_incr]:

BEGIN

IF .current\_stmt\_mode THEN current\_stmt = 1;

current\_incr = .dpc\_entry [next\_uns\_byte];

dpc\_entry = dpc\_entry [add\_two\_bytes];

END;

! Change the line increment from its present value to  
! the value contained in the next word.

[dst\$sk\_set\_linum\_incr\_w]:

BEGIN

IF .current\_stmt\_mode THEN current\_stmt = 1;

current\_incr = .dpc\_entry [next\_uns\_word];

dpc\_entry = dpc\_entry [add\_three\_bytes];

END;

! Revert to a line increment of value 1.

[dst\$sk\_reset\_linum\_incr]:

BEGIN

IF .current\_stmt\_mode THEN current\_stmt = 1;

current\_incr = 1;

dpc\_entry = dpc\_entry [add\_one\_byte];

END;

[dst\$sk\_beg\_stmt\_mode]:

792	0920	U
793	0921	U
794	0922	U
795	0923	U
796	0924	U
797	0925	U
798	0926	U
799	0927	U
800	0928	U
801	0929	U
802	0930	U
803	0931	U
804	0932	U
805	0933	U
806	0934	U
807	0935	U
808	0936	U
809	0937	U
810	0938	U
811	0939	U
812	0940	U
813	0941	U
814	0942	U
815	0943	U
816	0944	U
817	0945	U
818	0946	U
819	0947	U
820	0948	U
821	0949	U
822	0950	U
823	0951	U
824	0952	U
825	0953	U
826	0954	U
827	0955	U
828	0956	U
829	0957	U
830	0958	U
831	0959	U
832	0960	U
833	0961	U
834	0962	U
835	0963	U
836	0964	U
837	0965	U
838	0966	U
839	0967	U
840	0968	U
841	0969	U
842	0970	U
843	0971	U
844	0972	U
845	0973	U
846	0974	U
847	0975	U
848	0976	U

849	0977	4
850	0978	4
851	0979	4
852	0980	4
853	0981	4
854	0982	4
855	0983	4
856	0984	4
857	0985	3
858	0986	3
859	0987	3
860	0988	4
861	0989	4
862	0990	4
863	0991	4
864	0992	3
865	0993	3
866	0994	3
867	0995	4
868	0996	4
869	0997	4
870	0998	3
871	0999	3
872	1000	3
873	1001	4
874	1002	4
875	1003	4
876	1004	3
877	1005	3
878	1006	3
879	1007	4
880	1008	4
881	1009	4
882	1010	3
883	1011	3
884	1012	3
885	1013	4
886	1014	4
887	1015	4
888	1016	3
889	1017	3
890	1018	3
891	1019	4
892	1020	4
893	1021	4
894	1022	4
895	1023	4
896	1024	4
897	1025	4
898	1026	4
899	1027	3
900	1028	3
901	1029	3
902	1030	4
903	1031	4
904	1032	4
905	1033	4

```
BEGIN
IF .current_mark NEQ line_open
THEN
    SIGNAL(dbg$_invdstrec);

current_stmt = 1;
current_stmt_mode = TRUE;
dpc_entry = dpc_entry[add_one_byte];
END;

[dst$set_stmt_mode]:
BEGIN
current_stmt = 1;
current_stmt_mode = FALSE;
dpc_entry = dpc_entry[add_one_byte];
END;

[dst$set_linum_b]:
BEGIN
current_line = .dpc_entry[next_uns_byte];
dpc_entry = dpc_entry[add_two_bytes];
END;

[dst$set_linum]:
BEGIN
current_line = .dpc_entry[next_uns_word];
dpc_entry = dpc_entry[add_three_bytes];
END;

[dst$set_linum_l]:
BEGIN
current_line = .dpc_entry[next_uns_long];
dpc_entry = dpc_entry[add_five_bytes];
END;

[dst$set_stmtnum]:
BEGIN
current_stmt = .dpc_entry[next_uns_word];
dpc_entry = dpc_entry[add_three_bytes];
END;

[dst$set_pc]:
BEGIN
IF .current_mark NEQ line_closed
THEN
    SIGNAL (dbg$_invdstrec);

current_pc = .start_pc +
    .dpc_entry[next_uns_byte];
dpc_entry = dpc_entry[add_two_bytes];
END;

[dst$set_pc_w]:
BEGIN
IF .current_mark NEQ line_closed
THEN
    SIGNAL (dbg$_invdstrec);
```

906	1034	4
907	1035	4
908	1036	4
909	1037	4
910	1038	3
911	1039	3
912	1040	3
913	1041	4
914	1042	4
915	1043	4
916	1044	4
917	1045	4
918	1046	4
919	1047	4
920	1048	4
921	1049	3
922	1050	3
923	1051	3
924	1052	3
925	1053	3
926	1054	3
927	1055	4
928	1056	4
929	1057	4
930	1058	4
931	1059	4
932	1060	4
933	1061	4
934	1062	3
935	1063	3
936	1064	3
937	1065	4
938	1066	4
939	1067	4
940	1068	4
941	1069	4
942	1070	4
943	1071	3
944	1072	3
945	1073	3
946	1074	4
947	1075	4
948	1076	4
949	1077	4
950	1078	4
951	1079	4
952	1080	3
953	1081	3
954	1082	3
955	1083	3
956	1084	4
957	1085	4
958	1086	4
959	1087	4
960	1088	4
961	1089	4
962	1090	3

```
current_pc = .start_pc +  
                .dpc_entry[next_uns_word];  
dpc_entry = dpc_entry[add_three_bytes];  
END;  
[dst$sk_set_pc_l]:  
BEGIN  
IF .current_mark NEQ line_closed  
THEN  
    SIGNAL (dbg$_invdstrec);  
  
current_pc = .start_pc +  
                .dpc_entry[next_uns_long];  
dpc_entry = dpc_entry[add_five_bytes];  
END;  
  
! Set the current PC value to an absolute address.  
[dst$sk_set_abs_pc]:  
BEGIN  
IF .CURRENT_MARK NEQ LINE_CLOSED  
THEN  
    SIGNAL(DBG$_INV DSTREC);  
  
CURRENT_PC = .DPC_ENTRY[NEXT UNS LONG];  
DPC_ENTRY = DPC_ENTRY[ADD_FIVE_BYTES];  
END;  
[dst$sk_term]:  
BEGIN  
current_pc = .current_pc +  
                .dpc_entry[next_uns_byte];  
current_mark = line_closed;  
dpc_entry = dpc_entry[add_two_bytes];  
RETURN TRUE;  
END;  
[dst$sk_term_w]:  
BEGIN  
current_pc = .current_pc +  
                .dpc_entry[next_uns_word];  
current_mark = line_closed;  
dpc_entry = dpc_entry[add_three_bytes];  
RETURN TRUE;  
END;  
[dst$sk_term_l]:  
BEGIN  
current_pc = .current_pc +  
                .dpc_entry[next_uns_long];  
current_mark = line_closed;  
dpc_entry = dpc_entry[add_five_bytes];  
RETURN TRUE;  
END;
```

```
.. 963 1091 3
.. 964 1092 3
.. 965 1093 3
.. 966 1094 3
.. 967 1095 3
.. 968 1096 3
.. 969 1097 3
.. 970 1098 3
.. 971 1099 4
.. 972 1100 4
.. 973 1101 4
.. 974 1102 4
.. 975 1103 4
.. 976 1104 4
.. 977 1105 4
.. 978 1106 4
.. 979 1107 4
.. 980 1108 4
.. 981 1109 4
.. 982 1110 4
.. 983 1111 4
.. 984 1112 4
.. 985 1113 4
.. 986 1114 4
.. 987 1115 4
.. 988 1116 4
.. 989 1117 4
.. 990 1118 4
.. 991 1119 3
.. 992 1120 3
.. 993 1121 3
.. 994 1122 2
.. 995 1123 2
.. 996 1124 2
.. 997 1125 1
```

```
! This is a standard delta_PC command if the value is
! less than or equal to zero. Otherwise it is an error.
! If okay, set next_pc value, update the dpc_entry,
! and return with success.
```

```
[OUTRANGE]:
BEGIN
  IF .dpc_entry [current_byte] LSS
    dst$delta_pc_low
  OR .dpc_entry[current_byte] GTR
    dst$delta_pc_high
  THEN
    SIGNAL (dbg$invdstrec);

  IF .current_stmt_mode
  THEN
    current_stmt = .current_stmt + 1
  ELSE
    current_line = .current_line +
      .current_incr;

  current_pc = .current_pc -
    .dpc_entry [current_byte];
  current_mark = line_open;
  dpc_entry = dpc_entry [add_one_byte];
  RETURN TRUE;
END;
```

TES;

END;

RETURN 0;  
END;

63  
14

```
001C 0000U PROC_PC_CMD:
54 00000000G 00 9E 00002 .WORD Save R2,R3,R4 0806
53 00000000' EF 9E 00009 MOVAB LIB$SIGNAL, R4
50 FC B3 9A 00010 1$: MOVAB DPC_ENTRY, R3
50 FC A3 C0 00014 ADDL2 @DST_ENTRY, R0 0850
50 63 D1 00018 CMPL DST_ENTRY, R0
1B 15 0001B BLEQ DPC_ENTRY, R0 0849
50 A3 D6 0001D INCL PCTBL_COUNT
38 A3 50 A3 D1 00020 CMPL PCTBL_COUNT, NUM_PC_TBLS 0853
03 15 00025 BLEQ 2$ 0854
01ED 31 00027 BRW 56$
04 C0 0002A 2$: ADDL2 #4, CURRENT_TABLE 0855
FC A3 3C B3 D0 0002E MOVL @CURRENT_TABLE, DST_ENTRY 0856
FC A3 02 C1 00033 ADDL3 #2, DST_ENTRY, DPC_ENTRY 0857
52 63 D0 00038 3$: MOVL DPC_ENTRY, R2 0866
01 62 8F 0003B CASEB (R2), #1, #20
```

00C2  
0107  
0170  
018B  
012000A1  
00EE  
014F  
01B9  
0112008F  
00E0  
012E  
01A8  
00B30055  
00D1  
0119  
0127  
0075  
01C80003F 4\$:  
00047  
0004F  
00057  
0005F  
000678\$-4\$-  
16\$-4\$-  
17\$-4\$-  
21\$-4\$-  
23\$-4\$-  
25\$-4\$-  
27\$-4\$-  
29\$-4\$-  
33\$-4\$-  
38\$-4\$-  
41\$-4\$-  
45\$-4\$-  
36\$-4\$-  
51\$-4\$-  
52\$-4\$-  
47\$-4\$-  
13\$-4\$-  
19\$-4\$-  
31\$-4\$-  
34\$-4\$-  
53\$-4\$-

(R2)

TSTB

BLEQ

PUSHL

CALLS

BLBC

INCL

BRB

ADDL2

CVTBL

SUBL2

MOVL

INCL

BRB

BLBC

INCL

BRB

ADDL2

MOVL

MOVZWL

ADDL2

ADDL2

BRW

BLBC

INCL

BRB

ADDL2

MOVL

ADDL2

BRW

MOVZBL

ADDL2

BLBC

MOVL

BRB

BLBC

MOVL

5\$

#164650

#1, LIB\$SIGNAL

CURRENT\_STMT\_MODE, 6\$

CURRENT\_STMT

7\$

CURRENT\_INCR, CURRENT\_LINE

ADPC\_ENTRY, R0

R0, CURRENT\_PC

#1, CURRENT\_MARK

DPC\_ENTRY

12\$

CURRENT\_STMT\_MODE, 9\$

CURRENT\_STMT

10\$

CURRENT\_INCR, CURRENT\_LINE

#1, CURRENT\_MARK

1(R2), R0

R0, CURRENT\_PC

#3, DPC\_ENTRY

55\$

CURRENT\_STMT\_MODE, 14\$

CURRENT\_STMT

15\$

CURRENT\_INCR, CURRENT\_LINE

#1, CURRENT MARK

1(R2), CURRENT\_PC

54\$

1(R2), R0

R0, CURRENT LINE

CURRENT\_STMT\_MODE, 32\$

#1, CURRENT\_STMT

32\$

CURRENT\_STMT\_MODE, 18\$

#1, CURRENT\_STMT

1102

1105

1107

1109

1112

1115

1116

1117

1118

0876

0878

0881

0883

0885

0886

0887

0897

0899

0902

0904

0906

0907

0917

0918

0919

0928

08	50	01	A2	3C	000E8	18\$:	MOVZWL	1(R2), R0	0929
	A3		50	C0	000EC		ADDL2	R0, CURRENT_LINE	
			79	11	000F0		BRB	37\$	0930
	04	18	A3	E9	000F2	19\$:	BLBC	CURRENT_STMT_MODE, 20\$	0939
0C	A3		01	D0	000F6		MOVL	#1, CURRENT_STMT	
08	A3	01	A2	C0	000FA	20\$:	ADDL2	1(R2), CURRENT_LINE	0940
			63	11	000FF		BRB	35\$	0941
	04	18	A3	E9	00101	21\$:	BLBC	CURRENT_STMT_MODE, 22\$	0950
0C	A3		01	D0	00105		MOVL	#1, CURRENT_STMT	
10	A3	01	A2	9A	00109	22\$:	MOVZBL	1(R2), CURRENT_INCR	0951
			79	11	0010E		BRB	40\$	0952
	04	18	A3	E9	00110	23\$:	BLBC	CURRENT_STMT_MODE, 24\$	0961
0C	A3		01	D0	00114		MOVL	#1, CURRENT_STMT	
10	A3	01	A2	3C	00118	24\$:	MOVZWL	1(R2), CURRENT_INCR	0962
			4C	11	0011D		BRB	37\$	0963
	04	18	A3	E9	0011F	25\$:	BLBC	CURRENT_STMT_MODE, 26\$	0971
0C	A3		01	D0	00123		MOVL	#1, CURRENT_STMT	
10	A3		01	D0	00127	26\$:	MOVL	#1, CURRENT_INCR	0972
			20	11	0012B		BRB	30\$	0973
	01	1C	A3	D1	0012D	27\$:	CMPL	CURRENT_MARK, #1	0978
			09	13	00131		BEQL	28\$	
	0002832A		8F	DD	00133		PUSHL	#164650	0980
			01	FB	00139		CALLS	#1, LIB\$SIGNAL	
0C	A3		01	D0	0013C	28\$:	MOVL	#1, CURRENT_STMT	0982
18	A3		01	D0	00140		MOVL	#1, CURRENT_STMT_MODE	0983
			07	11	00144		BRB	30\$	0984
0C	A3		01	D0	00146	29\$:	MOVL	#1, CURRENT_STMT	0989
		18	A3	D4	0014A		CLRL	CURRENT_STMT_MODE	0990
			63	D6	0014D	30\$:	INCL	DPC_ENTRY	0991
			5C	11	0014F		BRB	44\$	0866
08	A3	01	A2	9A	00151	31\$:	MOVZBL	1(R2), CURRENT_LINE	0996
			31	11	00156	32\$:	BRB	40\$	0997
08	A3	01	A2	3C	00158	33\$:	MOVZWL	1(R2), CURRENT_LINE	1002
			4B	11	0015D		BRB	43\$	1003
08	A3	01	A2	D0	0015F	34\$:	MOVL	1(R2), CURRENT_LINE	1008
			7B	11	00164	35\$:	BRB	49\$	1009
0C	A3	01	A2	3C	00166	36\$:	MOVZWL	1(R2), CURRENT_STMT	1014
			3D	11	0016B	37\$:	BRB	43\$	1015
	02	1C	A3	D1	0016D	38\$:	CMPL	CURRENT_MARK, #2	1020
			09	13	00171		BEQL	39\$	
	0002832A		8F	DD	00173		PUSHL	#164650	1022
			01	FB	00179		CALLS	#1, LIB\$SIGNAL	
	64		63	D0	0017C	39\$:	MOVL	DPC_ENTRY, R0	1025
	50	01	A0	9A	0017F		MOVZBL	1(R0), R1	
14	51	04	B341	9E	00183		MOVAB	@START PC[R1], CURRENT_PC	1026
	A3		02	C0	00189	40\$:	ADDL2	#2, DPC_ENTRY	0866
	63		56	11	0018C		BRB	50\$	1031
		1C	A3	D1	0018E	41\$:	CMPL	CURRENT_MARK, #2	
			09	13	00192		BEQL	42\$	
	0002832A		8F	DD	00194		PUSHL	#164650	1033
			01	FB	0019A		CALLS	#1, LIB\$SIGNAL	
	64		63	D0	0019D	42\$:	MOVL	DPC_ENTRY, R0	1036
	50	01	A0	3C	001A0		MOVZWL	1(R0), R1	
14	51	04	B341	9E	001A4		MOVAB	@START PC[R1], CURRENT_PC	1037
	A3		03	C0	001AA	43\$:	ADDL2	#3, DPC_ENTRY	0866
	63		35	11	001AD	44\$:	BRB	50\$	1042
		1C	A3	D1	001AF	45\$:	CMPL	CURRENT_MARK, #2	

			0002832A	09	13	001B3	BEQL	46\$		
				8F	DD	001B5	PUSHL	#164650		1044
		64		01	FB	001BB	CALLS	#1, LIB\$SIGNAL		
14	A3	04	A3	01	63	D0 001BE	46\$: MOVL	DPC_ENTRY, R0		1047
					A0	C1 001C1	ADDL3	1(R0), START_PC, CURRENT_PC		
					17	11 001C8	BRB	49\$		1048
		02		1C	A3	D1 001CA	47\$: CMPL	CURRENT_MARK, #2		1056
					09	13 001CE	BEQL	48\$		
			0002832A		8F	DD 001D0	PUSHL	#164650		1058
		64			01	FB 001D6	CALLS	#1, LIB\$SIGNAL		
		50			63	D0 001D9	48\$: MOVL	DPC_ENTRY, R0		1060
		14	A3	01	A0	D0 001DC	MOVL	1(R0), CURRENT_PC		
			63		05	C0 001E1	49\$: ADDL2	#5, DPC_ENTRY		1061
					FE29	31 001E4	50\$: BRW	1\$		0866
		50		01	A2	9A 001E7	51\$: MOVZBL	1(R2), R0		1067
		14	A3		50	C0 001EB	ADDL2	R0, CURRENT_PC		
		1C	A3		02	D0 001EF	MOVL	#2, CURRENT_MARK		1068
			63		02	C0 001F3	ADDL2	#2, DPC_ENTRY		1069
					1B	11 001F6	BRB	55\$		1070
		50		01	A2	3C 001F8	52\$: MOVZWL	1(R2), R0		1076
		14	A3		50	C0 001FC	ADDL2	R0, CURRENT_PC		
		1C	A3		02	D0 00200	MOVL	#2, CURRENT_MARK		1077
					FEA7	31 00204	BRW	11\$		1078
		14	A3	01	A2	C0 00207	53\$: ADDL2	1(R2), CURRENT_PC		1086
		1C	A3		02	D0 0020C	MOVL	#2, CURRENT_MARK		1087
			63		05	C0 00210	54\$: ADDL2	#5, DPC_ENTRY		1088
			50		01	D0 00213	55\$: MOVL	#1, R0		1089
						04 00216	RET			
					50	D4 00217	56\$: CLRL	R0		1125
					04	00219	RET			

; Routine Size: 538 bytes, Routine Base: DBG\$CODE + 02A9

```

999 1126 1 ROUTINE FIND_EOL(LINE_END) =
1000 1127 1 ++
1001 1128 1 Functional description:
1002 1129 1 This routine processes PC correlation commands until
1003 1130 1 an end of line is found.
1004 1131 1
1005 1132 1 Inputs:
1006 1133 1 line_end - a copy-back pointer for the value of the end-of-line
1007 1134 1
1008 1135 1 Implicit inputs:
1009 1136 1 None
1010 1137 1
1011 1138 1 Implicit outputs:
1012 1139 1 the contents of the line pointer, the increment pointer, the
1013 1140 1 statement pointer, the next_pc pointer, dpc_entry, and possible
1014 1141 1 dst_entry are updated to new values.
1015 1142 1
1016 1143 1 Routine value:
1017 1144 1 TRUE if all goes well, otherwise FALSE.
1018 1145 1
1019 1146 1 Side effects:
1020 1147 1 More of the correlation records for this routine are read.
1021 1148 1 --
1022 1149 1
1023 1150 2 BEGIN
1024 1151 2
1025 1152 2 REPEAT
1026 1153 2 BEGIN
1027 1154 2
1028 1155 2
1029 1156 2 ! See whether the current record is exhausted. If
1030 1157 2 so, get a new record. If none are available,
1031 1158 2 return FALSE. Otherwise, set dpc_entry to point to
1032 1159 2 the address of the third byte of the correlation record.
1033 1160 2
1034 1161 2 IF dpc_entry[current_byte] GTR (.dst_entry[dst$b_length] +
1035 1162 2 dst_entry[dst$b_length])
1036 1163 2 THEN
1037 1164 2 BEGIN
1038 1165 2 PCTBL COUNT = .PCTBL COUNT + 1;
1039 1166 2 IF .PCTBL COUNT GTR .NUM_PC TBLS THEN RETURN FALSE;
1040 1167 2 current_table = .current_table + 4;
1041 1168 2 dst_entry = ..current_table;
1042 1169 2 dpc_entry = dst_entry[dst$b_vflags];
1043 1170 2 END;
1044 1171 2
1045 1172 2
1046 1173 2 ! Now process each command, either PC correlation or
1047 1174 2 delta-PC one at a time.
1048 1175 2
1049 1176 2 CASE .dpc_entry [current_byte] FROM 1 TO dst$k_pccor_high OF
1050 1177 2 SET
1051 1178 2 [dst$k_delta_pc_w]:
1052 1179 2 BEGIN
1053 1180 2 .line_end = (.current_pc - 1) +
1054 1181 2 .dpc_entry [next_uns_word];
1055 1182 2 RETURN TRUE;
```

1056	1183	W
1057	1184	W
1058	1185	W
1059	1186	W
1060	1187	W
1061	1188	W
1062	1189	W
1063	1190	W
1064	1191	W
1065	1192	W
1066	1193	W
1067	1194	W
1068	1195	W
1069	1196	W
1070	1197	W
1071	1198	W
1072	1199	W
1073	1200	W
1074	1201	W
1075	1202	W
1076	1203	W
1077	1204	W
1078	1205	W
1079	1206	W
1080	1207	W
1081	1208	W
1082	1209	W
1083	1210	W
1084	1211	W
1085	1212	W
1086	1213	W
1087	1214	W
1088	1215	W
1089	1216	W
1090	1217	W
1091	1218	W
1092	1219	W
1093	1220	W
1094	1221	W
1095	1222	W
1096	1223	W
1097	1224	W
1098	1225	W
1099	1226	W
1100	1227	W
1101	1228	W
1102	1229	W
1103	1230	W
1104	1231	W
1105	1232	W
1106	1233	W
1107	1234	W
1108	1235	W
1109	1236	W
1110	1237	W
1111	1238	W
1112	1239	W

```
END;
[dst$delta_pc_l]:
BEGIN
    .line_end = (.current_pc - 1) +
                .dpc_entry [next_uns_long];
RETURN TRUE;
END;

[dst$incr_linum]:
    dpc_entry = dpc_entry [add_two_bytes];

[dst$incr_linum_w]:
    dpc_entry = dpc_entry [add_three_bytes];

[dst$incr_linum_l]:
    dpc_entry = dpc_entry [add_five_bytes];

[dst$set_linum_incr]:
    dpc_entry = dpc_entry [add_two_bytes];

[dst$set_linum_incr_w]:
    dpc_entry = dpc_entry [add_three_bytes];

[dst$reset_linum_incr]:
    dpc_entry = dpc_entry [add_one_byte];

[dst$beg_stmt_mode]:
    dpc_entry = dpc_entry [add_one_byte];

[dst$end_stmt_mode]:
    dpc_entry = dpc_entry [add_one_byte];

[dst$set_linum_b]:
    dpc_entry = dpc_entry [add_two_bytes];

[dst$set_linum]:
    dpc_entry = dpc_entry [add_three_bytes];

[dst$set_linum_l]:
    dpc_entry = dpc_entry [add_five_bytes];

[dst$set_stmtnum]:
    dpc_entry = dpc_entry [add_three_bytes];

[dst$set_pc]:
BEGIN
    .line_end = (.start_pc - 1) +
                .dpc_entry [next_uns_byte];
RETURN TRUE;
END;

[dst$set_pc_w]:
BEGIN
    .line_end = (.start_pc - 1) +
                .dpc_entry [next_uns_word];
RETURN TRUE;
```

```
1113 1240
1114 1241
1115 1242
1116 1243
1117 1244
1118 1245
1119 1246
1120 1247
1121 1248
1122 1249
1123 1250
1124 1251
1125 1252
1126 1253
1127 1254
1128 1255
1129 1256
1130 1257
1131 1258
1132 1259
1133 1260
1134 1261
1135 1262
1136 1263
1137 1264
1138 1265
1139 1266
1140 1267
1141 1268
1142 1269
1143 1270
1144 1271
1145 1272
1146 1273
1147 1274
1148 1275
1149 1276
1150 1277
1151 1278
1152 1279
1153 1280
1154 1281
1155 1282
1156 1283
1157 1284
1158 1285
1159 1286
1160 1287
1161 1288
1162 1289
1163 1290
1164 1291
1165 1292
1166 1293
1167 1294
1168 1295
```

```
END;
[dst$sk_set_pc_l]:
BEGIN
    .line_end = (.start_pc - 1) +
                .dpc_entry[next_uns_long];
RETURN TRUE;
END;

[DST$K_SET_ABS_PC]:
BEGIN
    .LINE_END = .DPC_ENTRY[NEXT_UNLONG] - 1;
RETURN TRUE;
END;

[dst$sk_term]:
BEGIN
    .line_end = (.current_pc - 1) +
                .dpc_entry[next_uns_byte];
RETURN TRUE;
END;

[dst$sk_term_w]:
BEGIN
    .line_end = (.current_pc - 1) +
                .dpc_entry[next_uns_word];
RETURN TRUE;
END;

[dst$sk_term_l]:
BEGIN
    .line_end = (.current_pc - 1) +
                .dpc_entry[next_uns_long];
RETURN TRUE;
END;

[OUTRANGE]:
BEGIN
    IF .dpc_entry [current_byte] LSS
        dst$sk_delta_pc_low
    OR .dpc_entry[current_byte] GTR
        dst$sk_delta_pc_high
    THEN
        SIGNAL (dbg$_invdstrec);
    .line_end = (.current_pc - 1) -
                .dpc_entry [current_byte];
RETURN TRUE;
END;

TES;

END;

RETURN 0;
END;
```

```
000C 00000 FIND_EOL:
      53 00000000' EF 9E 00002 .WORD Save R2,R3          : 1126
      50 FC B3 9A 00009 1$: MOVAB DPC_ENTRY, R3          : 1162
      50 FC A3 C0 0000D ADDL2 DST_ENTRY, R0          : 1161
      50 63 D1 00011 CMPL DPC_ENTRY, R0          : 1165
      1B 15 00014 BLEQ 3$          : 1166
      50 A3 D6 00016 INCL PCTBL_COUNT          : 1167
      38 A3 50 A3 D1 00019 CMPL PCTBL_COUNT, NUM_PC_TBLS : 1168
      03 15 0001E BLEQ 2$          : 1169
      00AE 31 00020 BRW 22$          : 1176
      3C A3 04 C0 00023 2$: ADDL2 #4, CURRENT_TABLE
      FC A3 3C B3 D0 00027 MOVL @CURRENT_TABLE, DST_ENTRY
      FC A3 02 C1 0002C ADDL3 #2, DST_ENTRY, DPC_ENTRY
      52 63 D0 00031 3$: MOVL DPC_ENTRY, R2
      01 62 8F 00034 CASEB (R2), #1, #20
      0054 0046 00038 4$: .WORD 6$-4$,-
      0050 005E 00040 9$-4$,-
      0074 006A 00048 11$-4$,-
      007C 0084 00050 9$-4$,-
      0059 0054 00058 11$-4$,-
      008A 00060 8$-4$,-
      8$-4$,-
      8$-4$,-
      11$-4$,-
      13$-4$,-
      14$-4$,-
      16$-4$,-
      11$-4$,-
      18$-4$,-
      6$-4$,-
      17$-4$,-
      19$-4$,-
      10$-4$,-
      9$-4$,-
      10$-4$,-
      19$-4$
      62 95 00062 TSTB (R2)          : 1280
      0D 15 00064 BLEQ 5$          : 1283
      8F DD 00066 PUSHL #164650
      01 FB 0006C CALLS #1, LIB$SIGNAL          : 1286
      50 B3 98 00073 5$: CVTBL @DPC_ENTRY, R0
      4A 11 00077 SUBL3 R0, CURRENT_PC, R0
      50 4A 11 0007C BRB 20$          : 1285
      50 50 01 A2 3C 0007E 6$: MOVZWL 1(R2), R0
      50 50 14 A3 C0 00082 7$: ADDL2 CURRENT_PC, R0
      40 11 00086 BRB 20$          : 1180
      63 D6 00088 8$: INCL DPC_ENTRY          : 1214
      0D 11 0008A BRB 12$          : 1217
      63 02 C0 0008C 9$: ADDL2 #2, DPC_ENTRY
      08 11 0008F BRB 12$          : 1223
      63 05 C0 00091 10$: ADDL2 #5, DPC_ENTRY
      03 11 00094 BRB 12$
```

		63		03	C0	00096	11\$:	ADDL2	#3, DPC_ENTRY	:	1226	
				FF6D	31	00099	12\$:	BRW	1\$	:		
		50	01	A2	9A	0009C	13\$:	MOVZBL	1(R2), R0	:	1231	
				04	11	000A0		BRB	15\$	:		
		50	01	A2	3C	000A2	14\$:	MOVZWL	1(R2), R0	:	1238	
		50	04	A3	C0	000A6	15\$:	ADDL2	START_PC, R0	:		
				1C	11	000AA		BRB	20\$	:	1237	
	50	04	A3	01	A2	C1	000AC	16\$:	ADDL3	1(R2), START_PC, R0	:	1245
				14	11	000B2		BRB	20\$	:	1244	
04	BC	01	A2	01	C3	000B4	17\$:	SUBL3	#1, 1(R2), @LINE_END	:	1251	
				11	11	000BA		BRB	21\$	:	1252	
		50	01	A2	9A	000BC	18\$:	MOVZBL	1(R2), R0	:	1258	
				C0	11	000C0		BRB	7\$	:	1265	
	50	14	A3	01	A2	C1	000C2	19\$:	ADDL3	1(R2), CURRENT_PC, R0	:	1272
		04	BC	FF	A0	9E	000C8	20\$:	MOVAB	-1(R0), @LINE_END	:	1271
		50		01	D0	000CD	21\$:	MOVL	#1, R0	:	1273	
					04	000D0		RET		:		
				50	D4	000D1	22\$:	CLRL	R0	:	1295	
					04	000D3		RET		:		

; Routine Size: 212 bytes, Routine Base: DBG\$CODE + 04C3

```
1170 1296 1 ROUTINE GIVE_LINE_INFO(LINE_NUM, STMT_NUM): NOVALUE =
1171 1297 1
1172 1298 1 FUNCTION
1173 1299 1     This routine gives prev., current, next line information to the user
1174 1300 1     when the desired line is not found.
1175 1301 1
1176 1302 1 INPUTS
1177 1303 1     REPORT_PREV_LINE - Previous line
1178 1304 1     REPORT_PREV_STMT - Previous statement
1179 1305 1     LINE_NUM         - Current line
1180 1306 1     STMT_NUM         - Current statement
1181 1307 1     REPORT_NEXT_LINE - Next line
1182 1308 1     REPORT_NEXT_STMT - Next statement
1183 1309 1
1184 1310 1 OUTPUTS
1185 1311 1     Informational message is displayed. No return value.
1186 1312 1
1187 1313 1
1188 1314 2 BEGIN
1189 1315 2
1190 1316 2 LOCAL
1191 1317 2     BUFFER: VECTOR[80, BYTE];      ! Output buffer
1192 1318 2     BUF_DESC: VECTOR[2, LONG];    ! Output buffer string descriptor
1193 1319 2
1194 1320 2
1195 1321 2 IF .STMT_NUM EQL 0 THEN STMT_NUM = 1;
1196 1322 2 IF .REPORT_PREV_STMT EQL 0 THEN REPORT_PREV_STMT = 1;
1197 1323 2 IF .REPORT_NEXT_STMT EQL 0 THEN REPORT_NEXT_STMT = 1;
1198 1324 2
1199 1325 2 BUF_DESC[0] = 79;
1200 1326 2 BUF_DESC[1] = BUFFER[1];
1201 1327 2
1202 1328 2 IF (.REPORT_PREV_LINE EQL 0) AND
1203 1329 2     (.LINE_NUM EQL .REPORT_NEXT_LINE) AND
1204 1330 2     (.REPORT_PREV_STMT EQL 1) AND
1205 1331 2     (.STMT_NUM EQL .REPORT_NEXT_STMT)
1206 1332 2 THEN
1207 1333 2     BEGIN
1208 1334 2         DBG$FORMAT_FAO_OUT(BUF_DESC, UPLIT BYTE
1209 1335 2             (%ASCIC 'no line information available'));
1210 1336 2         BUFFER[0] = 79 - .BUF_DESC[0];
1211 1337 2         SIGNAL(DBG$LINEINFO, 1, BUFFER);
1212 1338 2         RETURN 0;
1213 1339 2     END;
1214 1340 2
1215 1341 2 DBG$FORMAT_FAO_OUT(BUF_DESC, UPLIT BYTE(%ASCIC 'no line !UL'), .LINE_NUM);
1216 1342 2 IF .STMT_NUM GTR 1
1217 1343 2 THEN
1218 1344 2     DBG$FORMAT_FAO_OUT(BUF_DESC, UPLIT BYTE(%ASCIC '!.!UL'), .STMT_NUM);
1219 1345 2
1220 1346 2 IF NOT (.REPORT_PREV_LINE EQL 0 AND .REPORT_PREV_STMT EQL 1)
1221 1347 2 THEN
1222 1348 2     BEGIN
1223 1349 2         DBG$FORMAT_FAO_OUT(BUF_DESC, UPLIT BYTE
1224 1350 2             (%ASCIC ', previous line is !UL'), .REPORT_PREV_LINE);
1225 1351 2
1226 1352 2     IF .REPORT_PREV_STMT GTR 1
```

```

1227      1353      1      THEN
1228      1354      1      DBGSFORMAT_FAO_OUT(BUF_DESC, UPLIT BYTE(%ASCIC '.!UL'), .REPORT_PREV_STMT);
1229      1355      1      END;
1230      1356      1
1231      1357      1      IF NOT (.REPORT_NEXT_LINE EQL .LINE_NUM AND
1232      1358      1      .REPORT_NEXT_STMT EQL .STMT_NUM)
1233      1359      1      THEN
1234      1360      1      BEGIN
1235      1361      1      DBGSFORMAT_FAO_OUT(BUF_DESC, UPLIT BYTE
1236      1362      1      (%ASCIC ', next line is !UL'), .REPORT_NEXT_LINE);
1237      1363      1
1238      1364      1      IF .REPORT_NEXT_STMT GTR 1
1239      1365      1      THEN
1240      1366      1      DBGSFORMAT_FAO_OUT(BUF_DESC, UPLIT BYTE(%ASCIC '.!UL'), .REPORT_NEXT_STMT);
1241      1367      1      END;
1242      1368      1
1243      1369      1      BUFFER[0] = 79 - .BUF_DESC[0];
1244      1370      1      SIGNAL(DBG$_LINEINFO, -1, BUFFER);
1245      1371      1      RETURN 0;
1246      1372      1      END;

```

```

.PSECT DBG$PLIT,NOWRT, SHR, PIC,0
6D 72 6F 66 6E 69 20 65 6E 69 6C 20 6F 6E 1D 00000 P.AAA: .ASCII <29>\no line information available\
65 6C 62 61 6C 69 61 76 61 20 6E 6F 69 74 61 0000F
    4C 55 21 20 65 6E 69 6C 20 6F 6E 0B 0001E P.AAB: .ASCII <11>\no line !UL\
6E 69 6C 20 73 75 6F 69 76 65 72 70 20 2E 04 0002A P.AAC: .ASCII <4>\.!UL\
    4C 55 21 20 73 69 20 2C 16 0002F P.AAD: .ASCII <22>\, previous line is !UL\
73 69 20 65 6E 69 6C 20 74 78 65 6E 20 2C 12 00046 P.AAE: .ASCII <4>\.!UL\
    4C 55 21 20 74 78 65 6E 20 2C 12 0004B P.AAF: .ASCII <18>\, next line is !UL\
    4C 55 21 2E 04 0005A
    4C 55 21 2E 04 0005E P.AAG: .ASCII <4>\.!UL\

```

```

.PSECT DBG$CODE,NOWRT, SHR, PIC,0
001C 00000 GIVE_LINE INFO:
54 00000000G 00 9E 00002 .WORD Save R2,R3,R4
53 00000000' EF 9E 00009 MOVAB DBGSFORMAT_FAO_OUT, R4
52 00000000' EF 9E 00010 MOVAB P.AAA, R3
5E A8 AE 9E 00017 MOVAB REPORT_PREV_STMT, R2
    08 AC D5 0001B MOVAB -88(SPT, SP)
    04 12 0001E TSTL STMT_NUM
08 AC 01 D0 00020 BNEQ 1$
    62 D5 00024 1$: MOVL #1, STMT_NUM
    03 12 00026 TSTL REPORT_PREV_STMT
    62 01 D0 00028 BNEQ 2$
    F8 A2 D5 0002B 2$: MOVL #1, REPORT_PREV_STMT
    04 12 0002E TSTL REPORT_NEXT_STMT
    F8 A2 01 D0 00030 BNEQ 3$
    04 6E 4F 8F 9A 00034 3$: MOVZBL #79, BUF_DESC
    AE 09 AE 9E 00038 MOVAB BUFFER+1, BUF_DESC+4

```

		FC	A2	D5	0003D	TSTL	REPORT_PREV_LINE	1328
			1D	12	00040	BNEQ	4\$	
F4	A2	04	AC	D1	00042	CMPL	LINE_NUM, REPORT_NEXT_LINE	1329
			16	12	00047	BNEQ	4\$	
	01		62	D1	00049	CMPL	REPORT_PREV_STMT, #1	1330
			11	12	0004C	BNEQ	4\$	
F8	A2	08	AC	D1	0004E	CMPL	STMT_NUM, REPORT_NEXT_STMT	1331
			0A	12	00053	BNEQ	4\$	
			53	DD	00055	PUSHL	R3	1334
		04	AE	9F	00057	PUSHAB	BUF_DESC	
	64		02	FB	0005A	CALLS	#2, -DBG\$FORMAT_FAO_OUT	
			70	11	0005D	BRB	9\$	1336
		04	AC	DD	0005F	PUSHL	LINE_NUM	1341
		1E	A3	9F	00062	PUSHAB	P.AAB	
		08	AE	9F	00065	PUSHAB	BUF_DESC	
	64		03	FB	00068	CALLS	#3, -DBG\$FORMAT_FAO_OUT	
	01	08	AC	D1	0006B	CMPL	STMT_NUM, #1	1342
			0C	15	0006F	BLEQ	5\$	
		08	AC	DD	00071	PUSHL	STMT_NUM	1344
		2A	A3	9F	00074	PUSHAB	P.AAC	
		08	AE	9F	00077	PUSHAB	BUF_DESC	
	64		03	FB	0007A	CALLS	#3, -DBG\$FORMAT_FAO_OUT	
	50	FC	A2	D0	0007D	MOVL	REPORT_PREV_LINE, R0	1346
			05	12	00081	BNEQ	6\$	
	01		62	D1	00083	CMPL	REPORT_PREV_STMT, #1	
			1B	13	00086	BEQL	7\$	
			50	DD	00088	PUSHL	R0	1350
		2F	A3	9F	0008A	PUSHAB	P.AAD	1349
		08	AE	9F	0008D	PUSHAB	BUF_DESC	
	64		03	FB	00090	CALLS	#3, -DBG\$FORMAT_FAO_OUT	
	01		62	D1	00093	CMPL	REPORT_PREV_STMT, #1	1352
			0B	15	00096	BLEQ	7\$	
			62	DD	00098	PUSHL	REPORT_PREV_STMT	1354
		46	A3	9F	0009A	PUSHAB	P.AAE	
		08	AE	9F	0009D	PUSHAB	BUF_DESC	
	64		03	FB	000A0	CALLS	#3, -DBG\$FORMAT_FAO_OUT	
	04	AC	A2	D1	000A3	CMPL	REPORT_NEXT_LINE, LINE_NUM	1357
			07	12	000A8	BNEQ	8\$	
	08	AC	A2	D1	000AA	CMPL	REPORT_NEXT_STMT, STMT_NUM	1358
			1E	13	000AF	BEQL	9\$	
		F4	A2	DD	000B1	PUSHL	REPORT_NEXT_LINE	1362
		4B	A3	9F	000B4	PUSHAB	P.AAF	1361
		08	AE	9F	000B7	PUSHAB	BUF_DESC	
	64		03	FB	000BA	CALLS	#3, -DBG\$FORMAT_FAO_OUT	
	01	F8	A2	D1	000BD	CMPL	REPORT_NEXT_STMT, #1	1364
			0C	15	000C1	BLEQ	9\$	
		F8	A2	DD	000C3	PUSHL	REPORT_NEXT_STMT	1366
		5E	A3	9F	000C6	PUSHAB	P.AAG	
		08	AE	9F	000C9	PUSHAB	BUF_DESC	
	64		03	FB	000CC	CALLS	#3, -DBG\$FORMAT_FAO_OUT	
08	AE	4F	6E	83	000CF	SUBB3	BUF_DESC, #79, -BUFFER	1369
			08	AE	9F	PUSHAB	BUFFER	1370
			01	DD	000D8	PUSHL	#1	
			8F	DD	000DA	PUSHL	#165635	
00000000G	00		03	FB	000E0	CALLS	#3, LIB\$SIGNAL	
			04	000E7	RET			1372

: Routine Size: 232 bytes, Routine Base: DBG\$CODE + 0597

: 1247 1373 1  
: 1248 1374 1 END  
: 1249 1375 0 ELUDOM

## .EXTRN LIB\$SIGNAL

## PSECT SUMMARY

Name	Bytes	Attributes
DBG\$OWN	88	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, CON, PIC, ALIGN(2)
DBG\$CODE	1663	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(0)
DBG\$PLIT	99	NOVEC, NOWRT, RD, EXE, SHR, LCL, REL, CON, PIC, ALIGN(0)

## Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	0	0	1000	00:01.8
-\$255\$DUA28:[DEBUG.OBJ]STRUCDEF.L32;1	32	0	0	7	00:00.1
-\$255\$DUA28:[DEBUG.OBJ]DBGLIB.L32;1	1545	56	3	97	00:01.8
-\$255\$DUA28:[DEBUG.OBJ]DSTRECRDS.L32;1	418	127	30	31	00:00.3
-\$255\$DUA28:[DEBUG.OBJ]DBGMSG.L32;1	386	2	0	22	00:00.3
-\$255\$DUA28:[DEBUG.OBJ]DBGGEN.L32;1	150	0	0	12	00:00.3

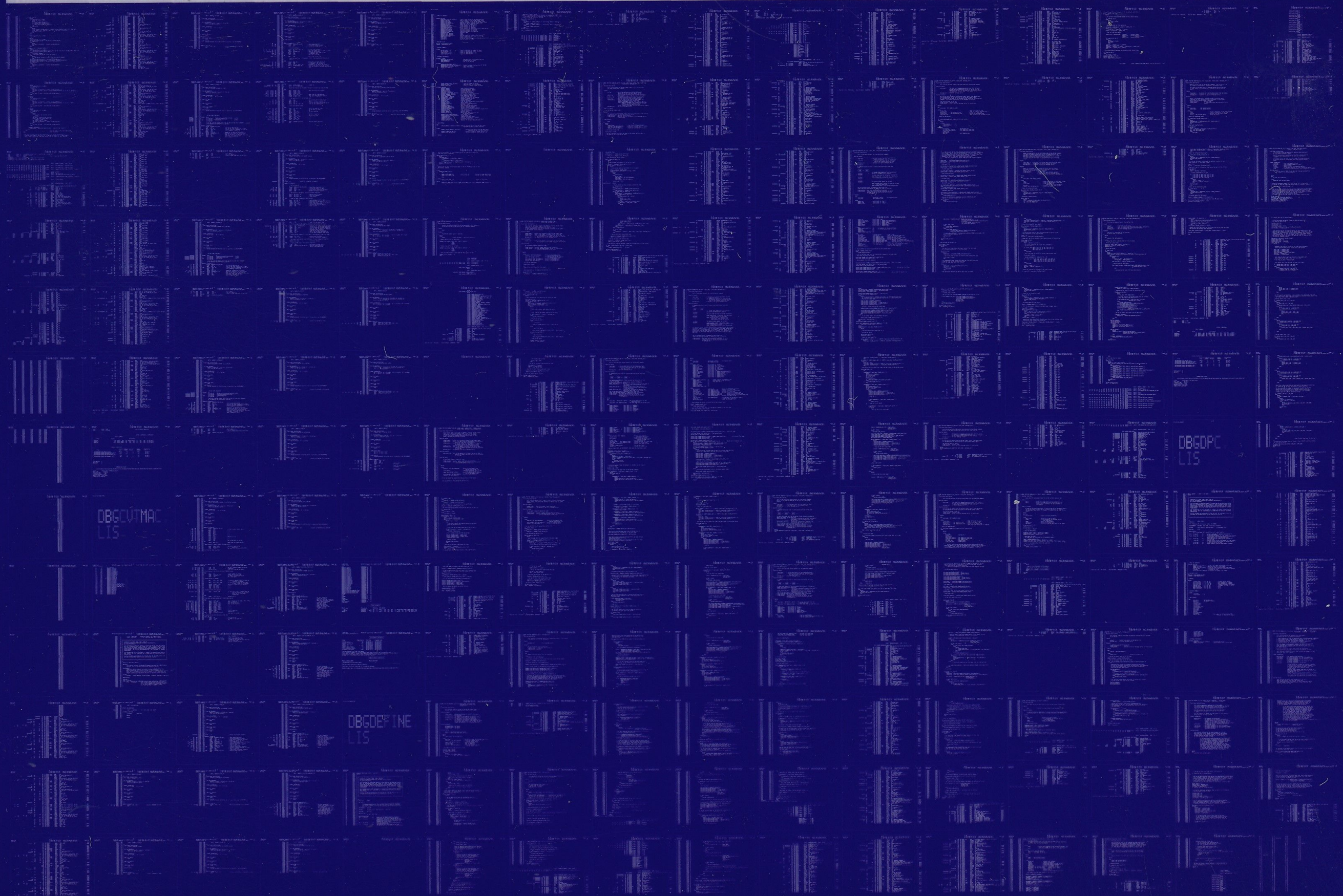
## COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:DBGDPC/OBJ=OBJ\$:DBGDPC MSRC\$:DBGDPC/UPDATE=(ENH\$:DBGDPC)

: Size: 1663 code + 187 data bytes  
: Run Time: 00:35.2  
: Elapsed Time: 02:05.1  
: Lines/CPU Min: 2343  
: Lexemes/CPU-Min: 12071  
: Memory Used: 221 pages  
: Compilation Complete

0079 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY



0080 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

